Intérprete para un lenguaje *A*

Especificación del lenguaje

Autor: Alberto Ruiz Andrés
Antonio Sanjuán de la Mano
Mario Villacorta García
Fecha de realización: 27 de mayo de 2020
Fecha de entrega: 31 de mayo de 2020
Valladolid

Universidad de Valladolid, Escuela de Ingeniería Informática
Gramáticas y lenguajes formales

Índice de Contenidos

1. Introducción **1**

2. Especificaciones léxicas **1**
2.1. Identificadores . . . 1
2.2. Comentarios . . . 1
2.3. Separadores . . . 1
2.4. Palabras clave . . . 2
2.5. Constantes . . . 2
2.5.1. Constantes numéricas . . . 2
2.5.2. Constantes lógicas . . . 3
2.5.3. Constantes carácter y tiras de caracteres . . . 3
2.6. Operadores . . . 3
2.6.1. Operadores unarios . . . 3
2.6.2. Operadores de asignación . . . 3
2.6.3. Operadores aritméticos . . . 3
2.6.4. Operadores lógicos . . . 4
2.6.5. Precedencia y asociatividad de los operadores . . . 5

3. Especificaciones sintácticas **5**
3.1. Sentencias . . . 5
3.1.1. Sentencias simples . . . 5
3.1.2. Sentencias bloque . . . 5
3.1.3. Sentencia de declaración . . . 5
3.1.4. Sentencia de asignación . . . 6
3.1.5. Sentencias de control . . . 6
3.1.5.1. Estructuras de decisión . . . 6
3.1.5.2. Estructuras de bucle . . . 6
3.2. Expresiones . . . 6
3.2.1. Expresiones unarias . . . 6
3.2.2. Expresiones aritméticas . . . 6
3.2.3. Expresiones lógicas . . . 7
3.3. Funciones predefinidas . . . 7
3.3.1. Funciones del sistema . . . 7
3.3.2. Funciones numéricas . . . 8

Anexos **9**

A. Metodología de trabajo **9**
1.1. Planteamiento . . . 9
1.2. Inicio del proyecto . . . 9
1.3. Finalización del proyecto . . . 9

B. Materiales complementarios **9**
2.1. Esquema de especificaciones 9
2.2. Especificación de identificadores 9
2.3. Precedencia y asociatividad de operadores 9

1. Introducción

El **lenguaje *A*** es una implementación simplificada del lenguaje de programación C. Se trata de un lenguaje de programación de alto nivel, interpretado, imperativo con sistema de tipos débil y estático. Está dedicado a las funciones básicas de todo lenguaje de programación.

2. Especificaciones léxicas

2.1. Identificadores

Los identificadores de *A* tienen las siguientes características: [1]

- Están formados por dígitos, letras y algunos signos de puntuación del juego ASCII, más concretamente:
  - Todos los caracteres alfanuméricos.
  - El carácter '_'.
- Comienzan por una letra o el carácter '_'.
- Son únicos y no son una palabra reservada.
- Se distingue entre mayúsculas y minúsculas (case-sensitive).

2.2. Comentarios

Encontramos dos tipos de comentario en *A*

- Comentarios de única línea: definidos a tras la construcción '//'.
- Comentarios multilínea: definidos entre '/*' y '*/'.

2.3. Separadores

Los separadores del lenguaje son:

- El espacio en blanco para las constantes, literales, expresiones y sentencias.
- El salto de linea para los comentarios.
- Los paréntesis para asociar expresiones.

2.4. Palabras clave

A continuación se presenta una tabla con las palabras reservadas del lenguaje *A* y su utilidad.

Palabra clave	**Descripción**
if	Hace referencia a la estructura de control que comprueba si se cumple la condición especificada.
else	Hace referencia a la sentencia alternativa a la condición de un bucle if.
do	Hace referencia a la sentencia principal de una estructura de bucle do ... while ... .
while	Hace referencia a la condición que seguirá el bucle anterior. Además hace referecia a un bucle formado unicamente por esta palabra reservada while ... .
printf	Hace referencia a la función de salida estándar.
scanf	Hace referencia a la función de entrada estándar.
false	Hace referencia al valor lógico false.
var	Hace referencia a la declaración de una variable.
true	Hace referencia al valor lógico true.
sin	Hace referencia a la operacion trigonométrica del seno.
cos	Hace referencia a la operacion trigonométrica del coseno.
tan	Hace referencia a la operacion trigonométrica de la tangente.
arcsin	Hace referencia a la operacion trigonométrica del arcoseno.
arccos	Hace referencia a la operacion trigonométrica del arcocoseno.
arctg	Hace referencia a la operacion trigonométrica de la arcotangente.
mcm	Hace referencia a el cálculo del mínimo común múltiplo de dos expresiones numéricas.
mcd	Hace referencia al cálculo del máximo común divisor de dos expresiones numéricas.
log	Hace referencia al cálculo del logaritmo de una expresión numérica y toma como base otra expresión numérica.

2.5. Constantes

2.5.1. Constantes numéricas

Las constantes númericas son enteras y reales implementadas a través de tipo flotante de doble precisión. Las constantes positivas no llevan ningún símbolo adicional y las negativas son antecedidas por el operador unario '-'.

Una constante numérica esta compuesta por una parte entera, el carácter '.' y una parte decimal. La parte entera es obligatoria, mientras que el carácter '.' y la parte decimal pueden aparecer o no en función de si la cantidad a representar es entera o no.

Tanto la parte entera como la parte decimal están constituidas por los dígitos del 0 al 9 con una o más apariciones.

2.5.2. Constantes lógicas

Las constantes lógicas se definen a través de los literales *true* y *false*, que representan los valores lógicos.

2.5.3. Constantes carácter y tiras de caracteres

Las constantes carácter corresponden a un único carácter del juego ASCII en entrecomillado simple. Su valor es interpretado como numérico.

Las tiras de caracteres son cualquier concatenación de constantes carácter en entrecomillado doble.

2.6. Operadores

2.6.1. Operadores unarios

El lenguaje define tres unarios:

- '-': Operador de negatividad de constantes numéricas.
- '++': Operador incremento unitario de constantes numéricas.
- '−−': Operador decremento unitario de constantes numéricas.
- '!': Operador de negación lógico.

2.6.2. Operadores de asignación

Los operadores de asignación implementados son 5:

- Asignación básica: mediante el operador '='. Se sitúa entre una variable previamente declarada y una expresión cuya evaluación será el nuevo valor.
- Asignaciones operacionales: mediante los operadores unarios '++' y '−−'.

2.6.3. Operadores aritméticos

Los operadores aritméticos se sitúan entre dos expresiones aritméticas. Son los siguientes:

- '+: implementa la operación suma.
- '-': implementa la operación resta.
- '*': implementa la operación producto.
- '/': implementa la operación división real.
- '**': implementa la operación potenciación.
- '%': implementa la operación modulo.
- '++': implementa la operación incremento.
- '−−': implementa la operación decremento.

2.6.4. Operadores lógicos

Los operadores lógicos se sitúan entre dos expresiones lógicas. Son los siguientes:

- '&': implementa la operación lógica AND.
- '|': implementa la operación lógica OR.
- '!': implementa la operación lógica NOT.
- '==': implementa la operación lógica de comparación de igualdad.
- '!=': implementa la operación lógica de comparación de desigualdad.
- '>' : implementa la operación lógica de comparación $>$.
- '≥': implementa la operación lógica de comparación $\geq$.
- '<': implementa la operación lógica de comparación $<$.
- '≤': implementa la operación lógica de comparación $\leq$.

2.6.5. Precedencia y asociatividad de los operadores

Tipo de operador	Operador	Nivel de precedencia	Asociatividad
Asignación	'='	8	derecha
	'++'	1	izquierda
	'−−'	1	izquierda
Aritméticos	'+'	3	izquierda
	'-'	3	izquierda
	'-' (unario)	1	derecha
	'*'	2	izquierda
	'/'	2	izquierda
	'%'	2	izquierda
	'**'	1	derecha
Lógicos	'|'	7	izquierda
	'&'	6	izquierda
	'!'	1	derecha
	'=='	5	izquierda
	'!='	5	izquierda
	'>'	4	izquierda
	'≥'	4	izquierda
	'<'	4	izquierda
	'≤'	4	izquierda

3. Especificaciones sintácticas

3.1. Sentencias

Las sentencias son las instrucciones que no devuelven valor o devuelven valor de tipo *void*. Se consideran sentencias tanto las sentencias simples como las sentencias bloque.

3.1.1. Sentencias simples

Las sentencias simples están constituidas por una instrucción terminada en ';'.

3.1.2. Sentencias bloque

Las sentencias bloque están constituidas por una sucesión de sentencias encerradas entre llaves '{' '}'.

3.1.3. Sentencia de declaración

Para declarar una variable de forma no tipada se sigue la estructura: *var <nombreVariable>*

3.1.4. Sentencia de asignación

- Asignación: Se almacena un valor en una variable. Su estructura es: *<variable> = <expresión>*.
- Expresiones unarias: incremento y decremento.

3.1.5. Sentencias de control

3.1.5.1. Estructuras de decisión

- Bifurcación simple: *if (<expresión lógica>) <sentencia>*
- Bifurcación doble: *if (<expresión lógica>) <sentencia bloque> else <sentencia bloque>*

3.1.5.2. Estructuras de bucle

- Bucle que se ejecuta al menos una vez: *do <sentencia bloque> while (<expresión lógica>)*
- Bucle condicional: *while (<expresión lógica>) <sentencia bloque>*

3.2. Expresiones

Las expresiones son las instrucciones que devuelven valor de algún tipo distinto de *void*. Se asocian mediante los caracteres de paréntesis '(', ')'.

3.2.1. Expresiones unarias

- Incremento: Se suma 1 al valor de una variable numérica y se le asigna ese nuevo valor. La estructura es: *<variable> ++*.
- Decremento: Se suma -1 al valor de una variable numérica y se le asigna ese nuevo valor. La estructura es: *<variable> –*.
- Cambio de signo: Para obtener el opuesto de una expresión, la estructura a seguir es: *- <expresion numérica>*. Devuelve el valor de la expresión numérica original multiplicado por -1.
- Negación: De un modo similar al opuesto, para la negación de una expresión, la estructura que se debe seguir es: *! <expresión lógica>*. Devuelve el valor contrario al de la expresión lógica original.

3.2.2. Expresiones aritméticas

Devuelven constantes numéricas reales de punto flotante de doble precisión.

- Suma: *<expresión numérica> + <expresión numérica>*.
- Resta: *<expresión numérica> - <expresión numérica>*.
- Multiplicación: *<expresión numérica> * <expresión numérica>*.

- División: *<expresión numérica> / <expresión numérica>*.
  - Error si la segunda expresión es 0.
- Módulo: *<expresión numérica> % <expresión numérica>*
  - Error si la segunda expresión es 0.
- Potenciación: *<expresión numérica> ** <expresión numérica>*
  - Error si se trata de *0 ** 0*.

3.2.3. Expresiones lógicas

Devuelven valores lógicos (*true* o *false*)

- Conjunción: *<expresión lógica> & <expresión lógica>*.
- Disyunción: *<expresión lógica> | <expresión lógica>*.
- Negación: *! <expresión lógica>*.
- Comparación de igualdad:*<expresión de tipo X> == <expresión de tipo X>*
- Comparación de desigualdad:*<expresión de tipo X> != <expresión de tipo X>*
- Comparación >:*<expresión numérica> > <expresión numérica>*.
- Comparación ≥:*<expresión numérica> ≥ <expresión numérica>*.
- Comparación <:*<expresión numérica> < <expresión numérica>*.
- Comparación ≤:*<expresión numérica> ≤ <expresión numérica>*.

3.3. Funciones predefinidas

3.3.1. Funciones del sistema

- Impresión por salida estándar:
  - *printf(<expresion>) : void*
- Lectura por entrada estándar:
  - *scanf(<expresión>) : void*

3.3.2. Funciones numéricas

- Logaritmo: *log(<expresión numérica>, <expresión numérica>) : <expresión numérica>*
  $logaritmo(a, b) \rightarrow log_a b$
  - Error si alguna de las expresiones es ≤ 0
- Máximo común divisor: *mcd(<expresión numérica>, <expresión numérica>) : <expresión numérica>*
  El resultado es el máximo valor que divide exactamente a ambas expresiones.
  - Error si alguna de las dos expresiones $\notin \mathbb{Z}^+$.
- Mínimo común múltiplo: *mcm(<expresión numérica>, <expresión numérica>) : <expresión numérica>*
  El resultado es un múltiplo común de ambas expresiones cuyo valor es el más pequeño posible.
  - Error si alguna de las dos expresiones $\notin \mathbb{Z}^+$.
- Funciones trigonométricas:
  - Seno: *sin(<expresión numérica>) : <expresión numérica>*
    La expresión de entrada representa un ángulo en radianes.
  - Coseno: *cos(<expresión numérica>) : <expresión numérica>*
    La expresión de entrada representa un ángulo en radianes.
  - Tangente: *tan(<expresión numérica>) : <expresión numérica>*
    La expresión de entrada representa un ángulo en radianes.
    - Error si la expresión es $\frac{\pi}{2} + k\pi, k \in \mathbb{Z}$.
  - Arcoseno: *arcsin(<expresión numérica>) : <expresión numérica>*
    - Error si la expresión de entrada es >1.
  - Arcocoseno: *arccos(<expresión numérica>) : <expresión numérica>*
    - Error si la expresión de entrada es >1.
  - Arcotangente: *arctg(<expresión numérica>) : <expresión numérica>*

Anexos

A. Metodología de trabajo

Para la realización de la tarea el trabajo se ha dividido fundamentalmente en tres días.

1.1. Planteamiento

Los principios del lenguaje se plantearon el día 14 de mayo de 2020. Entre todos los componentes se definió la funcionalidad a implementar a través de la conferencia grupal en el Campus Virtual.

1.2. Inicio del proyecto

El proyecto se inició el 18 de mayo de 2020 a través de la plataforma colaborativa Overleaf por Mario Villacorta. Se definieron las especificaciones léxicas iniciales en un documento LaTeX.

1.3. Finalización del proyecto

El proyecto se finalizó el día 31 de mayo de 2020. Se definieron las especificaciones sintácticas por Alberto Ruiz y Antonio Sanjuán y se repasó entre todos los componentes del grupo el documento completo.

B. Materiales complementarios

2.1. Esquema de especificaciones

El esquema de las especificaciones léxicas y sintácticas del lenguaje se ha basado en el presentado para Java en *Java 7: Los fundamentos del lenguaje Java, ediciones ENI marzo 2012* de Thierry Groussard y para C en *C/C++: Curso de programación 2015, Anaya 2014* de Miguel Ángel Acera.

2.2. Especificación de identificadores

La especificación de los identificadores del lenguaje *A* está basada en la de Java.
https://javadesdecero.es/fundamentos/identificadores-y-palabras-reservadas/

2.3. Precedencia y asociatividad de operadores

La precedencia y la asociatividad de los operadores está basada en la propia del lenguaje C presentada en los siguientes enlaces:

- https://es.wikipedia.org/wiki/Anexo:Operadores_de_C_y_C++
- https://docs.microsoft.com/es-es/cpp/c-language/precedence-and-order-of-evaluation?view=vs-2019

Referencias

[1] C. Pes, “Identificadores en lenguaje c.” Online. http://www.carlospes.com/curso_de_lenguaje_c/01_05_identificadores.php.