UNIVERSIDAD
DE GRANADA

Práctica 1:
Diseño del Lenguaje

Procesadores de Lenguajes

(Grupo A2, Equipo 7)

2024-2025

Joaquín Arcila Pérez

Gerardo Arenas Nasrawin

Jaime Martínez Bravo

Álvaro Molina Álvarez

Índice

1. Introducción **1**

2. Descripción formal de la sintaxis del lenguaje usando BNF **2**

3. Definición de la semántica **5**

4. Tabla de tokens **7**

1. Introducción

El lenguaje asignado tiene los siguientes elementos:

1. Sintaxis inspirada en **lenguaje C**
2. Palabras reservadas en **Inglés**
3. Estructura de datos considerada como tipo elemental: **listas**
4. Subprogramas: **funciones**
5. Estructuras de control adicional: **do-until**

2. Descripción formal de la sintaxis del lenguaje usando BNF

```
# Descripción de elementos básicos
<letra>          ::= A..Z
                   | a..z
<digito_no_0>    ::= 1..9
<digito>         ::= <digito_no_0>
                   | 0
<numero_no_cero>::= <digito_no_0> | <numero><digito>
<entero>         ::= <numero_no_cero> | 0
<entero_signo>   ::= <signo><entero> | <entero>
<signo>          ::= +
                   | -
<bool>           ::= TRUE
                   | FALSE
<cadena>         ::= <letra>
                   | <numero>
                   | <cadena><cadena>
                   | <cadena>_<cadena>
<cad_num>        ::= <numero>
                   | <numero>.<parte_decimal>
                   | <numero_signo>
                   | <numero_signo>.<parte_decimal>
<parte_decimal>  ::= <digito><parte_decimal>
                   | <digito>
<identificador>  ::= <letra>
                   | <letra><cadena>

# Tipos de variables
<tipo_variable_elemental>    ::= int
                                | float
                                | char
                                | bool

# Operadores
<operador_monario>   ::= !
<operador_binario>   ::= +
                       | -
                       | *
                       | /
                       | %
                       | ==
```

```
                        | !=
                        | <
                        | >
                        | <=
                        | >=
                        | &
                        | $
                        | ?

# Expresiones
<exp_simple>    ::= <cad_num>
                  | <letra>
                  | <bool>
<exp>           ::= <operador_monario> <exp>
                  | <exp><operador_binario><exp>
                  | (<exp>)
                  | <exp_simple>

# Programa
<programa>  ::= int main <bloque>
<bloque>    ::= {
                    <declar_de_variables_locales>
                    <declar_de_funciones>
                    <sentencias>
                };

# Declaración variables locales
<declar_de_variables_locales> ::= local {
                                      <variables_locales>
                                  }
<variables_locales>           ::= <variables_locales> <cuerpo_declar_variables>
                                  | <cuerpo_declar_variables>
<cuerpo_declar_variables>     ::= <tipo_variable> <variables>;
<variables>                   ::= <identificador>, <variables>
                                  | <identficador>

# Declaración subprogramas (funciones)
<declar_de_funciones>   ::= <declar_de_funciones> <declar_de_funcion>
                          | <declar_de_funcion>
<declar_de_funcion>     ::= <cabecera_funcion> <bloque>
<cabecera_funcion>      ::= <tipo_variable> <identificador> ( <parametros> )
<parametros>            ::= <parametros>, <parametro>
                          |  <parametro>
<parametro> ::= <tipo_variable> <identificador>
```

```
85
86
87  # Sentencias
88  <sentencias>            ::= <sentencias> <sentencia>;
89                            | <sentencia>;
90  <sentencia>             ::= <bloque>
91                            | <sentencia_asignacion>
92                            | <sentencia_if>
93                            | <sentencia_while>
94                            | <sentencia_entrada>
95                            | <sentencia_salida>
96                            | <sentencia_return>
97                            | <sentencia_do_until>
98  <sentencia_asignacion>  ::= <identificador> = <exp>
99                            | list <identificador> = [<items>]
100 <items>                 ::= <exp>
101                           | <exp>,<items>
102 <sentencia_if>          ::= if (<exp>) <bloque>
103                           | if (<exp>) <bloque> else <bloque>
104 <sentencia_while>       ::= while (<exp>) <bloque>
105 <sentencia_entrada>     ::= cin >> <datos_entrada>
106 <datos_entrada>         ::= <exp> >> <datos_entrada>
107                           | <exp>
108 <sentencia_salida>      ::= cout << <datos_salida>
109 <datos_salida>          ::= <dato_salida> << <datos_salida>
110                           | <dato_salida>
111 <dato_salida>           ::= <exp>
112                           | <cad>
113 <sentencia_return>      ::= return <exp>
114 <sentencia_do_until>    ::= do <bloque> until (<exp>)
115
```

3. Definición de la semántica

1. **Elementos básicos**
   - **Letra:** Representa cualquier carácter alfabético, ya sea en mayúsculas (A-Z) o minúsculas (a-z).
   - **Dígito:** Es un número del 0 al 9. Si es distinto de 0, se llama <digito_no_0>.
   - **Número:** Puede ser un solo dígito o una cadena de dígitos, comenzando por un dígito distinto de 0.
   - **Número con signo:** Representa cualquier número que puede ir precedido de un signo (+ o -) o no.
   - **Cadena:** Es una secuencia de letras, números o combinaciones de ambas, pudiendo también contener guiones bajos.
   - **Cadena de números:** Es una secuencia de números con o sin signo, con la opción de añadir un punto seguido de un número, de forma que se puedan añadir decimales.
   - **Identificador:** Es una cadena forzada a empezar por una letra.
2. **Tipos de variables**
   - **int:** Entero (números sin decimales).
   - **float:** Números con decimales.
   - **char:** Un carácter.
   - **bool:** Representa valores lógicos ("TRUE" o "FALSE").
   - **list:** Representa listas de valores.
3. **Operadores**
   - **Monarios:** Representa los operadores que actúan sobre un sólo operando.
   - **Binarios:** Representa los operadores que actúan sobre dos operandos:
     - Aritméticos (+, -, *, /, %): Suman, restan, multiplican, dividen y calculan el resto de una división.
     - Relacionales (==, !=, <, >, <=, >=): Comparan valores, devolviendo TRUE o FALSE.
     - Lógicos (&, $, ?): Operaciones lógicas AND, OR y XOR respectivamente.
4. **Expresiones**
   - **Simples**: se traducen en valores de distintos tipos:

- Números (<cad_num>).
- Caracteres (<letra>).
- Valores lógicos (<bool>).

- **Compuestas**: se descomponen en expresiones simples o anidadas dentro de paréntesis.

5. **Programa**

   Todo programa empieza con la función principal `int main`, cuyo cuerpo principal está contenido en un bloque <bloque>.

6. **Declaración de variables locales**

   Se definen dentro de un bloque local usando su tipo ("`int`", "`float`", "`char`", "`bool`" o "`list`") seguido de los nombres de las variables, usando " , " como separador en caso de querer definir más de una variable en una misma orden, y acabando en " ; ".

7. **Declaración de subprogramas: funciones**

   Las funciones se definen indicando el tipo de variable de retorno, el nombre de la función (<identificador>), una lista de parámetros (entre paréntesis, usando como separador " , ") y el cuerpo de la función (<bloque>).

8. **Sentencias**

   - **Bloque:** Comienza con el símbolo de inicio " { "; contiene la declaración de variables locales, la declaración de funciones y las sentencias; y termina con el símbolo de fin " } ".
   - **Asignación:** Para dar valores a una variable se debe indicar el nombre de la variable, seguido de " = ", y una expresión en el caso de variables elementales; o una sucesión de valores, separados por " , " y entre corchetes en el caso de las listas.
   - **Estructuras de control:** "`if-then-else`", "`while`" o "`do-until`".
   - **Entrada y salida:** Sirven para leer valores del usuario o imprimir resultados en pantalla. Para la entrada se usa "`cin`" separando las expresiones con "`>>`", y análogamente para la salida se usará "`cout`" con "`<<`" como separador. Ambas permiten todos los tipos elementales, pero las cadenas de caracteres sólo se permiten en la salida.
   - **Return:** Finaliza una función y devuelve un valor. Se indica con `return` seguido de una expresión <exp>.

4. Tabla de tokens

Token	Código	Atributos	Patrón
ID	257	Lexema	([A-Z]|[a-z])([A-Z]|[a-z]|[0..9]|_)*
WHILE	258		"while"
DO	259		"do"
UNTIL	260		"until"
PYC	261		";"
PARIZQ	262		"("
PARDCH	263		")"
SUMREST	264	0:+, 1:-	("+"|"-")
OPEMON	265		"!"
OPEBIN	266	Lexema	("*"|"/"|"%"|"=="|"!="|"<"|">"| |"<=" |">="|"&"|"$")
ASING	267		"="
TIPOVAR	268	0:float, 1:char 3:bool	("float"|"char"|"bool")
TIPOLISTA	269		"list"
CORIZQ	270		"["
CORDCH	271		"]"
PUNTO	272		"."
COMA	273		","
SEPENT	274		">>"
SEPSAL	275		"<<"
VALBOOL	276	0:TRUE, 1:FALSE	("TRUE"|"FALSE")
CIN	277		"cin"
COUT	278		"cout"
MAIN	279		"main"
RETURN	280		"return"
LLAVEIZQ	281		"{"
LLAVEDCH	282		"}"
IF	283		"if"
ELSE	284		"else"
LOCAL	285		"local"
INT	286		"int"