UNIVERSIDAD DE LAS PALMAS DE GRAN CANARIA
Escuela de Ingeniería Informática

Procesadores de Lenguaje:

Nombre de nuestro lenguaje

Curso 2018 / 2019

Realizado por:
Medina Medina, David Alberto
Brito Ramos, Christian
Hernández Delgado, Christopher
López González, Néstor

Índice

1. Definición del lenguaje (Autor: Quien termine antes)

Introducir breve introducción del lenguaje que planteamos.

1.1. Tipos de datos (David)

Cualquier leguaje de programación necesita definir un conjunto de *tipos de datos*, esto es, la batería de valores y operaciones que puede adquirir una variable. Cada tipo de dato está definido en el lenguaje por un *literal* único que lo representa, lo que permite que cada tipo de dato tenga un representación física específica.
Los tipos de datos definidos en el lenguaje son los que figuran en el *cuadro 1*. Las características críticas de implementación que define a cada tipo son:

Entero Representa a todas y cada una de las variables enteras que sean declaradas en el lenguaje. Este tipo de dato presenta un tamaño de 4 bytes (32 bits) y permite representar números enteros con signo. El *complemento a 2* es el sistema elegido para definir el signo del número entro. Este dato se representa por el literal `int`. El rango de valores que puede tomar es

$$\left[-2^{N-1},\ 2^{N-1}-1\right] = \left[-2^{32-1},\ 2^{32-1}-1\right] = [-2147483648,\ 2147483647] \tag{1}$$

donde, N es el número de bits disponibles para representar el número el número entero (32 bits).

Coma flotante de simple precisión Este tipo de dato representa número reales en coma flotante de simple precisión con un tamaño de 4 bytes (32 bits) siguiendo el estándar *IEEE 754*. En la figura 1.1 puede observarse como esta representación binaria los bits se organizan en tres sectores principales:

- **Signo** (1 bit). Se trata de un sólo bit que define el signo del número: positivo (0) o negativo (1).
- **Exponente (8bits)**. Se trata de un número entero con signo de complemento a 2 ($[-128,\ 127]$)
- **Mantisa (23 bits)**. Conforma la fracción a la derecha de la coma binaria y un bit de encabezado implícito.

Este tipo de dato se representa con el literal `real`. Su rango de valores es de $\left[1{,}18 \cdot 10^{-38},\ 3{,}4 \cdot 10^{38}\right]$.

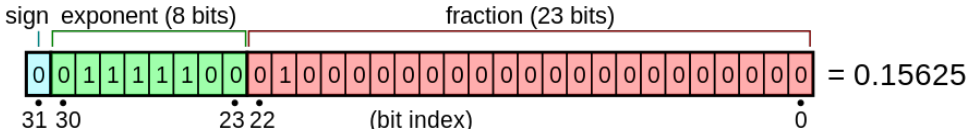

Figura 1: Representación binaria de número en coma flotante de simple precisión (*IEEE 754*)

Caracter Este tipo de dato es usado para representar caracteres de codificación *UTF-8*, es por este motivo que el tamaño de que ocupan las variables de tipo caracter son de 4 bytes siendo `char` el literal que lo representa.

Booleano Se trata de un tipo de dato utilizado para representar representar valores booleanos. Su tamaño es de 1 bit, por lo que tan solo puede tomar dos valores: `1` (verdadero) ó `0` (falso). El literal que lo representa es `bool`.

Cuadro 1: Tipos de datos

Tipo	**Literal**	**Tamaño**	**Rango**
Entero	int	4 Bytes	$[-2147483648,\ 2147483647]$
Coma flotante de simple precisión	real	4 Byte	$[1{,}18 \cdot 10^{-38},\ 3{,}4 \cdot 10^{38}]$
Caracter	char	1 Byte	*UTF-8*
Lógico	bool	1 Bit	$[0,\ 1]$

En el siguiente ejemplo se muestra cómo se declaran las variables con los literales de los tipos descritos anteriormente:

```
int entero      .. Esto es un entero
real flotante   .. Esto es un numero en coma flotante
char caracter   .. Esto es un caracter
bool booleano   .. Esto es un booleano
```

1.2. Colecciones de datos: `Arrays`

Las variables pueden ser agrupadas en colecciones de datos de una dimensión denominados *arrays*. En este lenguaje, cualquier tipo de dato puede formar parte de un *array*.

1.3. Palabras reservadas (Christian)

Aquí va el texto. Poner siempre un código de ejemplo.

1.4. Comentarios (Christian)

Aquí va el texto. Poner siempre un código de ejemplo.

1.5. Tipos de operadores

Aquí va el texto. Poner siempre un código de ejemplo.

1.5.1. Operadores aritméticos (David)

Aquí va el texto. Poner siempre un código de ejemplo.

1.5.2. Operadores lógicos (Néstor)

Aquí va el texto. Poner siempre un código de ejemplo.

1.5.3. Operadores bit a bit (Christian)

Aquí va el texto. Poner siempre un código de ejemplo.

1.5.4. Operadores de array (Christopher)

Aquí va el texto. Poner siempre un código de ejemplo.

1.6. Estructuras de control

Aquí va el texto. Poner siempre un código de ejemplo.

1.6.1. Sentencias `if-ifelse-else` (Néstor)

Aquí va el texto. Poner siempre un código de ejemplo.

1.6.2. Bucle `for-forelse-else` (Christopher)

Aquí va el texto. Poner siempre un código de ejemplo.

1.6.3. Bucle `while-whileelse-else` (Christian)

Aquí va el texto. Poner siempre un código de ejemplo.

1.7. Funciones (David)

Aquí va el texto. Poner siempre un código de ejemplo.

1.8. Funciones primitivas (Néstor)

Aquí va el texto. Poner siempre un código de ejemplo.

1.9. Código ejemplo (Christopher)

Aquí va el código de ejemplo con el que probaremos nuestro compilador.