**Parcial Compiladores**

**Docente:**

**Gabriel Taborda Blandón**

**Código CPI82-2**

**Fecha 2023-2**

**Integrantes:**

**Juan Camilo Ocampo Gomez**

**Juan Esteban Álvarez Ruiz**

**Johan Alexander Sánchez Aguirre**

**Lenguaje de programación**

Un lenguaje de programación es un lenguaje formal (o artificial, es decir, un lenguaje con reglas gramaticales bien definidas) que proporciona a una persona, en este caso el programador, la capacidad y habilidad de escribir (o programar) una serie de instrucciones o secuencias de órdenes en forma de algoritmos con el fin de controlar el comportamiento físico o lógico de un sistema informático, para que de esa manera se puedan obtener diversas clases de datos o ejecutar determinadas tareas. A todo este conjunto de órdenes escritas mediante un lenguaje de programación se le denomina programa informático.

**Tipo básicos de lenguajes por generaciones.**

Primera generación (década de 1940):

**Lenguaje de máquina**: Los primeros lenguajes de programación eran códigos binarios directamente legibles por las computadoras. Los programadores tenían que escribir instrucciones en código de máquina, lo que era tedioso y propenso a errores.

Segunda generación (década de 1950 – 1960):

**Lenguaje ensamblador**: Se introdujeron lenguajes ensambladores que permitían a los programadores usar mnemotecnia y abstracciones para escribir programas en lugar de código de máquina directo. Esto mejoró la legibilidad y la productividad.

Tercera generación (década de 1960-1970):

**Lenguajes de alto nivel**: Se desarrollaron lenguajes de programación de alto nivel como COBOL, Fortran, BASIC y C. Estos lenguajes eran más legibles, portátiles y fáciles de aprender en comparación con el ensamblador y el código de máquina. También introdujeron la abstracción de hardware.

Cuarta generación (década de 1970-1980):

**Lenguajes de programación de cuarta generación (4GL):** Estos lenguajes estaban orientados a aplicaciones específicas, como bases de datos y procesamiento de datos empresariales. Los ejemplos incluyen SQL y herramientas de desarrollo rápido de aplicaciones (RAD).

Quinta generación (década de 1980 en adelante):

**Lenguajes de programación lógica y de inteligencia artificial**: La quinta generación se centró en lenguajes de programación para la inteligencia artificial y la lógica, como Prolog y Lisp. Estos lenguajes se utilizan para resolver problemas complejos de forma declarativa.

**Componentes básicos de un lenguaje de programación:**

Un lenguaje de programación consta de varios componentes básicos que les permiten a los programadores escribir y ejecutar código de manera efectiva. Aquí están los componentes esenciales de un lenguaje de programación:

1. **Sintaxis**: Define la estructura y el formato que deben seguir las instrucciones en el código. Es crucial seguir la sintaxis correcta para que el programa se ejecute correctamente.
2. **Semántica**: Define el significado de las instrucciones y cómo interactúan entre sí. Incluye reglas y comportamientos del lenguaje.
3. **Variables y Tipos de Datos**: Los lenguajes de programación permiten a los programadores crear variables que pueden almacenar datos. Los tipos de datos determinan qué tipo de valores puede contener una variable (como números, texto, booleanos, etc.).
4. **Operadores**: Los operadores permiten manipular y operar sobre los datos. Esto puede incluir operaciones matemáticas, comparaciones, operaciones lógicas, etc.
5. **Estructuras de Control**: Estas permiten controlar el flujo de ejecución del programa. Incluyen estructuras como bucles (loops), condicionales (if-else), y switch-case.
6. **Estructuras de Datos:** Los lenguajes proporcionan formas de organizar y almacenar datos. Esto puede incluir listas, arreglos, conjuntos, diccionarios, entre otros.
7. **Funciones o Métodos**: Permiten a los programadores encapsular un conjunto de instrucciones y reutilizarlo a lo largo del programa. También facilitan la modularidad y mantenimiento del código.
8. **Entrada/Salida (E/S):** Proporciona las herramientas para interactuar con el mundo exterior, como recibir datos del usuario y mostrar resultados.
9. **Manejo de Excepciones**: Permite al programa gestionar errores y situaciones excepcionales de manera controlada.
10. **Bibliotecas o Módulos**: Proporcionan conjuntos de funciones y herramientas predefinidas que los programadores pueden utilizar para tareas comunes, como operaciones matemáticas avanzadas, manipulación de archivos, interacción con redes, etc.
11. **Comentarios**: Los comentarios permiten a los programadores incluir notas y explicaciones en el código fuente para que otros (y ellos mismos en el futuro) puedan entender el propósito y funcionamiento del código.
12. **Manejo de Memoria**: En lenguajes de bajo nivel, es importante gestionar la asignación y liberación de memoria manualmente. En lenguajes de alto nivel, esto suele ser administrado por el entorno de ejecución.
13. **Paradigmas de Programación**: Determina la filosofía y el enfoque que un lenguaje sigue para resolver problemas. Los paradigmas incluyen la programación imperativa, orientada a objetos, funcional, lógica, entre otros.
14. **Herramientas de Desarrollo**: Compiladores o intérpretes que traducen el código escrito por el programador en instrucciones entendibles por la máquina.

**Definición de un lenguaje de programa propio (Melody)**

**Introducción y descripción del lenguaje**

Nuestro objetivo es desarrollar un nuevo lenguaje de programación enfocado a personas relacionadas con el mundo musical, que además deseen desarrollar nuevas habilidades en el mundo de la programación y de esta forma tengan la capacidad de introducirse mucho más fácil a el mundo del desarrollo de software.

**Tipos básicos de datos o variables**

Los tipos de datos en este caso se representarán en base a la escala musical

int -> DO

float -> RE

double -> MI

string -> FA

char -> SOL

bool -> LA

object -> SI

**Operadores**

ASIGNAR (=)

JUNTAR à +

SEPARAR à -

AUMENTAR à \*

DISMINUIR à /

IGUAL à ==

DIFERENTE à ¡=

CONJUNTO à &&

ORQUESTA à ||

MAS\_ALTO\_QUE à >

MAS\_BAJO\_QUE à <

ALTO\_O\_IGUAL\_A à >=

BAJO\_O\_IGUAL\_A à >=

**Especificación para nombre identificadores**

Los identificadores deben tener al menos un carácter y pueden ser de longitud arbitraria.

Caracteres no permitidos:

No puede contener palabras reservadas como ejemplo no podría contener nada de la escala musical DO, RE, MI, FA, SOL, LA, SI.

No se permite espacio entre variables.

Los identificadores no pueden contener caracteres especiales como @, $ o %.

Mayúsculas y minúsculas:

Los identificadores son sensibles a mayúsculas y minúsculas. “Do” y “do” se considerarán identificadores distintos.

**Declaración de variables**

Para realizar una declaración valida primero se debe anteponer el tipo de dato basado en la escala musical DO, RE, MI, FA, SOL, LA ejemplo:

Para la declaración de un tipo de dato string sería de la siguiente manera:

FA Nombre ASIGNAR b\_Jorge\_b

Para la declaración de un tipo de dato numérico o lógico sería de la siguiente manera:

DO contadorUsuario ASIGNAR 5

**Constantes de cadena o cadena de caracteres**

Para las cadenas de caracteres utilizamos lo que en notación musical se conoce como un bemol, y se representa con la letra b, para nuestro caso le agregamos el guion bajo para saber desde dónde inicia y donde termina, ejemplo:   
FA letraCancion ASIGNAR b\_Hello World song\_b

**Comentarios**

Para los comentarios se utilizará la alteración musical Sostenido (#) permitido solo a nivel de línea ejemplo:

#Este código se encuentra comentado.

**Instrucción de asignación**

En este caso sería la palabra “ASIGNAR” ejemplo:

variable ASIGNAR valor

**Instrucciones de entrada y salida**

Musica.Escuchar(): Esta instrucción se utiliza para leer una línea de texto ingresada por el usuario.

Musica.Recitar(): Esta instrucción se utiliza para escribir texto.

**Instrucciones de decisión**

Para este caso se utilizará las dinámicas musicales las cuáles indican el nivel de sonoridad de una canción.

Para el caso del condicional de decisión “if” sería la palabra f(forte): Fuerte y para el caso del “else” sería ff(fortissimo): Muy fuerte ejemplo:

f(true)

SOL mivariable ASIGNAR 1

ff

SOL mivariable ASIGNAR 0

Para el tipo de condicional CASE sería la palabra reservada p(piano): suave

P(expresion)

Case valor1:

#Código a ejecutar 1

Case valor2:

#Código a ejecutar 1

**Instrucciones de repetición o ciclos**

Usando uso de componentes electrónicos se pueden hacer ciclos de música para que se repita de manera automática, esta técnica la denominamos LOOP y lo utilizaremos para los ciclos, ejemplo:

LOOP (estructura)

**Delimitadores de programa**

Se va a utilizar la palabra compas que es una unidad de medida de tiempo que contiene un número específico de pulsaciones

Compas

(Codigo)

Compas

**Ejemplo para el uso de instrucciones de decisión:**

Musica.Recitar(b\_ Por favor, ingresa tu edad: \_b)

DO edad ASIGNAR Musica.Escuchar()

f (edad > 18)

Compas

Musica.Recitar(b\_ Eres mayor de edad. Puedes ingresar. \_b)

Compas

ff

Compas

Musica.Recitar(b\_ Eres menor de edad. No puedes ingresar. \_b)

Compas

**Ejercicio aritmético:**

Este programa solicita al usuario que ingrese dos números, luego realiza varias operaciones aritméticas

con estos dos números los cuáles serán la división, multiplicación, resta y división,

al finalizar con la división se pedirá un tercer número para realizar el promedio entre estos tres números.

#Declaramos el método calcular con la palabra clave Melodia

Melodia calcular

compas

#Declaramos las variables tipo float

RE numero1

RE numero2

RE numero3

Musica.Recitar(b\_Por favor ingrese el primer número para el cálculo\_b)

numero1 ASIGNAR Musica.Escuchar()

Musica.Recitar(b\_Por favor ingrese el segundo número para el cálculo\_b)

numero2 ASIGNAR Musica.Escuchar()

#Suma

Musica.Recitar(b\_El resultado de la suma para los números ingresados es:\_b)

Musica.Recitar(numero1 JUNTAR numero2)

#Resta

Musica.Recitar(b\_El resultado de la suma para los números ingresados es:\_b)

Musica.Recitar(numero1 SEPARAR numero2)

#Multiplicación

Musica.Recitar(b\_El resultado de la suma para los números ingresados es:\_b)

Musica.Recitar(numero1 AUMENTAR numero2)

#División

Musica.Recitar(b\_El resultado de la suma para los números ingresados es:\_b)

Musica.Recitar(numero1 DISMINUIR numero2)

Musica.Recitar(b\_Ingrese un tercer número para obtener el promedio\_b)

numero3 ASIGNAR Musica.Escuchar()

#Promedio

Musica.Recitar(b\_El promedio para los tres números ingresados es:\_b)

Musica.Recitar( (numero1 JUNTAR numero2 JUNTAR numero3) DISMINUIR 3 )

Compas

**Ejercicio Ciclo**

Longitud de una cadena

Musica.Recibir(b\_Por favor ingrese el texto:\_b)

FA texto ASIGNAR Musica.Recibir

Do Contador ASIGNAR 0

LOOP( SOL caracter IN texto)

Compas

Contador ASINGAR Contador JUNTAR 1

Compas

Musica.Recitar(Contador)

Conclusiones personales

Hacer uso de los lenguajes de programación actuales tales de alto nivel como Java, C# o incluso JavaScript en muchas ocasiones te hacen olvidar de la verdadera complejidad que hay detrás de cada línea de código ya que la buena estructura, documentación y lo intuitivo que pueden llegar a hacer te facilitan muchas cosas.  
  
Al momento que quieres empezar a diseñar tu propio lenguaje entiendes que hay mucho más de fondo, muchos detalles a tener en cuenta que por pequeño que sea puede hacer una gran diferencia ya que no lo diseñas solo para ti sino para el mundo.

Bibliografía

<https://es.wikipedia.org/wiki/Lenguaje_de_programaci%C3%B3n>

<https://cv.uoc.edu/moduls/XW02_79049_00373/web/main/m4/v2_2.html>