Jorge Adrian Besnier Benavides

A01039882

LENGUAJES Y TRADUCTORES

Proyecto Final

**ESPECIFICACIONES GENERALES DEL PROYECTO**

**Profesora:** Dra. Norma Frida Roffe

**Objetivo del proyecto**: Crear un lenguaje de programaciónque tenga algunas instrucciones similares a dos de los siguientes tres lenguajes: Go, R, Ruby (en el diseño del lenguaje debe mostrar en forma clara cuáles son las similitudes).

**Propósito:** Constituir una herramienta de programación simple, con orientación a cálculos numéricos, útil para un programador principiante.

**INSTRUCCIONES:** El proyecto se desarrollará en forma individual. Para esto será necesario que cada uno diseñe los elementos de léxico, las estructuras sintácticas, reglas semánticas de traducción y la ejecución para un lenguaje que cumpla con los requisitos.

Contents

[**TITULO DEL PROGRAMA** 2](#_Toc50073196)

[Nombre 2](#_Toc50073197)

[Orientación 2](#_Toc50073198)

[Variables 2](#_Toc50073199)

[**ESTATUTOS** 1](#_Toc50073200)

[Definición de variables 1](#_Toc50073201)

[Asignación de expresiones a variables 1](#_Toc50073202)

[Lectura de variables 1](#_Toc50073203)

[Escritura de variables y strings 1](#_Toc50073204)

[Condicional 1](#_Toc50073205)

[Dos Ciclos Condicionales (tipo while) 1](#_Toc50073206)

[Un Ciclo Controlado: (tipo for) 1](#_Toc50073207)

[Llamadas a módulos sin paso de parámetros 1](#_Toc50073208)

[**MÓDULOS** 2](#_Toc50073209)

[Módulos/Funciones del código 2](#_Toc50073210)

[Usted defina las palabras reservadas 2](#_Toc50073211)

[Los operadores permitidos en las expresiones 2](#_Toc50073212)

[Operadores Relacionales 2](#_Toc50073213)

[Reglas de Expresiones 3](#_Toc50073214)

[Prioridad de operadores (mas alta a la mas baja) 3](#_Toc50073215)

[Realizar comentarios en el programa. 3](#_Toc50073216)

[**TIPOS DE DATOS** 4](#_Toc50073217)

[Simples 4](#_Toc50073218)

[Compuestos 4](#_Toc50073219)

[Todas las variables son globales 4](#_Toc50073220)

[**PUNTOS EXTRAS** 1](#_Toc50073221)

[LaPlace 1](#_Toc50073222)

[**Demonstración** 2](#_Toc50073223)

# **TITULO DEL PROGRAMA**

## Nombre

Mushin Programing Language

## Orientación

Cálculos Numéricos, para programadores principiantes

## Variables

Como es orientado a matemáticas se contará con variables del tipo

* Números sin decimales o bien “**Enteros**”
* Números con decimales o bien “**Flotantes**”
  + Hasta tres unidades después del punto.

**Enteros**

123

5/5 = Entero

9000

**Flotantes**

12.5

5/7 = flotante

0.553

Donde se busca que el compilador determine el tipo de variable si contiene un “.” en la notación, de igual manera se busca integrar una función que permita al programador cambiar entre los tipos de variables. Para poder hacer operaciones según sea necesario y según la tarea. Por otra parte, se busca integrar un tipo de dato que puede hacer operaciones con puntos flotantes y enteros sin preocupación.

* Garantizar Eficiencia y desempeño del código
* Consumir menos memoria si es posible

**Pasar de Int a float**

Agrega punto decimal

X = float(x)

**Pasar de float a int**

Pierde decimales

X= Int(x)

# **ESTATUTOS**

## Definición de variables

Para continuar con la idea de mantener la facilidad para el programador será posible que las variables sean definidas según sea necesario. Evitando que el usuario tenga que definir con anterioridad la variable. Sin embargo, para evitar que se asigne memoria a errores de dedo, etc. siempre la definición de una variable deberá de cumplir con:

**Correcto**

Variabe = dato que se quiera asignar

Variable = 1

X = 123.5

X = 0

**Incorrecto**

Variable =

Variable

X =

X

## Asignación de expresiones a variables

Para asignar una expresión ser hará siguiendo el mismo formato para la definición de variables solamente que con “:” para indicar que sigue una expresión. Asimismo, en este caso las variables dentro de la expresión deberán de estar previamente definidas antes de hacer uso de la variable con la expresión.

**Correcto**

a = 7

b = 8

x: a + b

**incorrecto**

a = 8

b = 7

x = a + b

## Lectura de variables

Se continuará con el mismo tipo de asignación para las variables solamente ahora se solicitará al usuario ingresar un valor. Además, se agregará un campo para agregar un mensaje de tal manera que la sintaxis es de la siguiente manera:

**Para leer una variable ingresada por el usuario**

X = input(“mensaje”)

**Para imprimir el valor de una variable**

print (x “mensaje” y “mensaje”)

## Escritura de variables y strings

Las variables deben de cumplir con el siguiente formato:

* Iniciar con una letra.
* Se admiten números y letras.
* Puede existir el guion bajo, pero no al final del nombre de la variable.
* No se admiten caracteres especiales.

**Correcto**

Jorge

Jorge\_123

Jorge123

**Incorrecto**

\_jorge123

123\_Jorge

Jorge123\_

En el caso de los strings que en este caso serán únicamente utilizados en la impresión de los datos deberá de cumplir con el siguiente formato:

* Iniciar con “ y terminar con “.
* Escribir el mensaje dentro de las comillas.
* Todo lo que se escribe adentro es impresión directa no puede tomar valores de las variables.

## Condicional

**Caso 1**

If **condicional**: Si el condicional es verdadero ejecutar este código, si no continuar con el código.

**Caso 2**

If **condicional**: Si el condicional es verdadero ejecutar este código.

else **condicional**: si el condicional es verdadero ejecutar este código.

else: si no se cumple nada ejecutar este código.

## Dos Ciclos Condicionales (tipo while)

**Caso 2**

While **condición**:

do: código principal que se intenta realizar

fail: código encaso que no se pueda ejecutar el do

else: código o break para continuar

**Caso 2**

cont = valor

While cont **condicon** valor:

do: Código a realizar

update: cont = acción para modificar el valor orginal

## Un Ciclo Controlado: (tipo for)

For controlVariable = valor; controlVariable condición valor; controlVariable updatea:

Código con tabulación para identificar el código dentro del ciclo

Se realiza n veces hasta que se cumpla la condición

## Llamadas a módulos sin paso de parámetros

**Para definir un modulo sin variable de entrada**

Def nombre ():

Código

Return valor que quiera

**Para llamar un módulo sin variable de entrada**

call nombre ()

# **MÓDULOS**

**Instrucciones:** Los módulos tendrán un nombre…. pero no parámetros ni variables locales. Pueden localizarse antes o después del programa principal. En un programa puede haber de 0 a n módulos. Las llamadas a módulos deben localizarse en los estatutos.

## Módulos/Funciones del código

**Para definir un módulo sin variable de entrada**

Def nombre ():

Código

Return para regresar al código

**Para llamar un módulo sin variable de entrada**

call nombre ()

**Para identificar el módulo principal**

Begin:

Código con tabulación indica código principal

## Usted defina las palabras reservadas

|  |  |
| --- | --- |
| **Palabra Reservada** | **Utilidad de la palabra reservada** |
| **Int()** | Transformar valores flotantes en valores sin punto decimal |
| **Float()** | Transformar valores enteros en valores con punto decimal |
| **Input()** | Solicitar al usuario que ingrese un valor externo |
| **Print()** | Imprimir los valores de las variables |
| **“ ”** | Indicar dentro del print que es un string y se debe de imprimir tal cual |
| **If** | Condicional que si se cumple una condición realizar determinado código |
| **While** | Un tipo de ciclo que espera que se cumpla determinada condición |
| **For** | Ciclo iterativo que actualiza un valor hasta que se cumple la condición de salida. |
| **Def** | Para definir funciones/módulos que realizan una acción repetitiva |
| **Call** | Llamar dentro del main a las funciones fuera del programa principal |
| **Begin** | Da inicio al código principal del programa. |

## Los operadores permitidos en las expresiones

|  |  |
| --- | --- |
| **Operador** | **Utilidad del operador** |
| **( )** | Ayuda a realizar un orden de operaciones para llegar al resultado deseado |
| **^** | Elevar a la potencia un numero o variable asimismo utilizar paréntesis para inticar que todo lo que sigue es parte de la potencia. Ejemplo: d^(a+5) |
| **\*** | Indicar la multiplicación entre dos valores o variables |
| **/** | Indicar la división entre dos valores o variables |
| **+** | Indicar la suma entre dos valores o variables |
| **-** | Indicar la resta ente dos valores o variables |

## Operadores Relacionales

|  |  |
| --- | --- |
| **Operador** | **Utilidad del operador** |
| < | Verdadero que el valor de la izquierda es menor que el valor a la derecha |
| > | Verdadero que el valor de la izquierda es mayor que el valor a la derecha |
| <= | Verdadero que el valor de la izquierda es menor o igual que el valor a la derecha |
| => | Verdadero que el valor de la izquierda es mayor o igual que el valor a la derecha |
| < > | Indica si los valores son diferentes |
| == | Indica si los valores son iguales (tener cuidado con el = simple de asignación) |

## Reglas de Expresiones

1. Solo los paréntesis pueden ser consecutivos para indicar el orden de operaciones (anidado dos)
2. Salvo el paréntesis no se permite que existan dos operadores seguidos como ++
3. No se permiten operadores relacionales seguidos

## Prioridad de operadores (mas alta a la mas baja)

|  |  |  |
| --- | --- | --- |
| **Prioridad** | **Operador** | **Condición** |
| 1 | ( ) | Del paréntesis más externo al paréntesis más interno. |
| 2 | ^ | Exponencial normal solo considerar que se puede utilizar el paréntesis para indicar que todo un conjunto forma parte del exponente |
| 3 | \* , / | Si se encuentren en el mismo nivel o paréntesis orden de izquierda a derecha |
| 4 | +,- | Si se encuentren en el mismo nivel o paréntesis orden de izquierda a derecha |
| 5 | Relacionales | Si se encuentren en el mismo nivel o paréntesis orden de izquierda a derecha |
| 6 | Not | No se cumple con la condición **(operación lógica y palabra reservada)** |
| 7 | And | Se cumple una y la otra condición **(operación lógica y palabra reservada)** |
| 8 | Or | Se cumple una o la otra condición **(operación lógica y palabra reservada)** |

\*nota los símbolos o palabras en las tablas también son palabras o orden de signos reservada.

## Realizar comentarios en el programa.

Se permite hacer el comentado del código con la secuencia de simbolos // como se muestra a continuación:

**Comentario en el programa**

//comentario en el código no se considera como algo que se tienen que ejecutar

# **TIPOS DE DATOS**

## Simples

**Instrucciones**: Al menos dos tipos de datos, usted decida si se deben declarar las variables o no.

Como es orientado a matemáticas se contará con variables del tipo

* Números sin decimales o bien “**Enteros**”
* Números con decimales o bien “**Flotantes**”
  + Hasta tres unidades después del punto.

**Enteros**

123

5/5 = Entero

9000

**Flotantes**

12.5

5/7 = flotante

0.553

**Asignación Correcta**

Variabe = dato que se quiera asignar

Variable = 1

X = 123.5

X = 0

**Asignación Incorrecta**

Variable =

Variable

X =

X

**Pasar de Int a float**

Agrega punto decimal

X = float(x)

**Pasar de float a int**

Pierde decimales

X= Int(x)

## Compuestos

**Instrucciones**: Vectores, Matrices y Cubos (*Variables dimensionadas de 1, 2 y 3 dimensiones*). Debe haber una declaración para este tipo de variables, para definir el tamaño de las dimensiones.

**Instrucciones:** La declaración de las dimensiones es numérica, pero la referencia es a través de expresiones aritméticas, por ejemplo, debe ser válido algo así como Matriz(i+1,i-1).

Siguiendo las reglas de asignación correctas previamente mencionadas para definir un conjunto de valores especial como vectores, matrices y cubos la asignación deberá de hacerse de la siguiente manera:

**Definir el tamaño de un cubo**

X = mat (Horizontal, Vertical, Profundidad)

**Para acceder a un valor del cubo**

Y = X(i, j, k)

**Para modificar un valor del cubo**

X(i, j, k) = 5

**Definir el tamaño de una matriz**

X = mat (Horizontal, Vertical)

**Para acceder a un valor del cubo**

Y = X(i, j)

**Para modificar un valor del cubo**

X(i, j) = 5

**Definir el tamaño de un vector**

X = mat (Horizontal)

**Para acceder a un valor del cubo**

Y = X(i)

**Para modificar un valor del cubo**

X(i) = 5

## Todas las variables son globales

Las variables creadas en el código de cualquier tipo, las palabras reservadas, operadores, etc. se pueden utilizar en todo el programa tanto en los módulos/funciones o en programa principal.

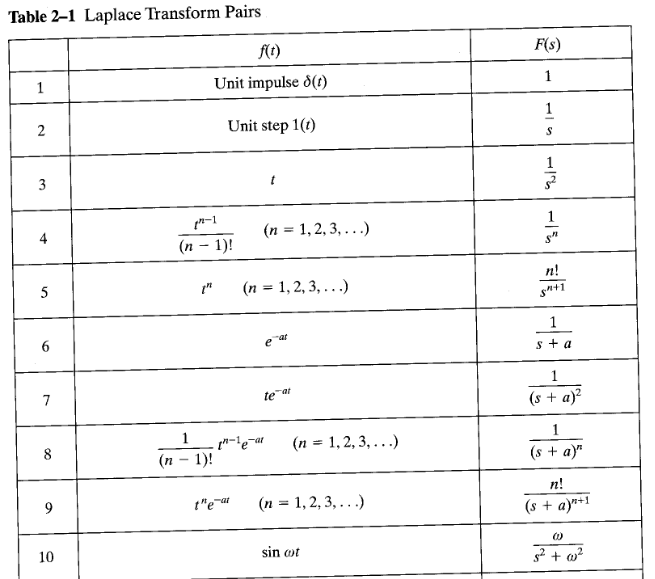
# **PUNTOS EXTRAS**

**Instrucciones** Se podrán obtener puntos extras cuando el lenguaje agregue alguna facilidad nueva (que usted invente) o añada una aplicación particular.

**Instrucciones** Que soporte aplicaciones matemáticas específicas (ejemplo Solución de Sistemas de Ecuaciones, Integración Numérica, etc.).

## LaPlace

**Transformar del domino del tiempo al domino de la frecuencia/dominio s, utilizar las tablas de conversión directa para las determinadas estructuras básicas en el dominio del tiempo como lo son las siguientes tabla:**



# **Demonstración**

**Instrucciones**: Utilizando su lenguaje, diseñe y muestre un programa que lea las dimensiones de dos matrices de tamaño máximo de 5x5, y las lea. El programa deberá contar con un menú y Utilice módulos

**Instrucciones**: Permite al usuario El usuario elige la opción de:

1. Multiplicar las dos matrices y dejar el resultado en una tercera matriz, imprimir resultado. Antes deberá verificar que sea posible realizar la multiplicación
2. Sumar las dos matrices y dejar el resultado en una tercera matriz, imprimir resultado. Antes deberá verificar que sea posible realizar la suma.

Def

Def

Begin: