Carlos Romero 11711209

Richardson Laínez 11711075

Julio Marin 10911010



Contenido

[Tipos de Variables y su Declaración 2](#_Toc52111365)

[Ejemplos de declaración y asignación de variables 2](#_Toc52111366)

[Main 3](#_Toc52111367)

[Print 3](#_Toc52111368)

[Input 3](#_Toc52111369)

[Comentarios 3](#_Toc52111370)

[Estructuras de Control 4](#_Toc52111371)

[**IF y SWITCH: una estructura unida** 4](#_Toc52111372)

[IF Con Comportamiento IF-ELSE 4](#_Toc52111373)

[IF Con Comportamiento SWTCH-CASE 6](#_Toc52111374)

[**FOR** 8](#_Toc52111375)

[Ejemplo de FOR 8](#_Toc52111376)

[**WHILE** 8](#_Toc52111377)

[Ejemplo de WHILE 8](#_Toc52111378)

[**FUNCIONES Y PROCEDIMIENTOS** 9](#_Toc52111379)

[Ejemplos de Función y Procedimiento 10](#_Toc52111380)

[Arreglos Unidimensionales y Bidimensionales 11](#_Toc52111381)

# Tipos de Variables y su Declaración

Los mencionados en la tabla son los tipos de datos que Rill-20 admite y con los que se puede trabajar.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Nombre de Tipo | | Tipo que representa | | Ejemplo de declaración | |
| int | | INTEGER: Números Enteros | | int x.,  int x:0.,  int x,y,z., | |
| bln | | BOOLEAN: True o Flase | | bln x.,  bln x:true.,  bln x:false.,  bln x,y,x., | |
| chr | | CHARACTER: Caracter | | chr x.,  chr x:’a’.,  chr x,y,x., | |
| Arr | | ARRAY: Arreglos Unidimensionales | | Se habla de estos más adelante. | |
| Mtx | | MATRIX: Arreglos Bidimensionales | | Se habla de estos más adelante. | |

La asignación de valores en le variable se hace con el símbolo “:” (dos puntos).

## Ejemplos de declaración y asignación de variables

**int** entero: 2020 .,

**int** a, b, c .,

**bln** aprobado: **true** .,

**chr** letra\_alfabeto: ‘a’ .,

Nota: La declaración y ejemplo de ello para los Arreglos Unidimensionales y Bidimensionales se muestra en la sección de [Arreglos Unidimensionales y Bidimensionales](#_Arreglos_Unidimensionales_y).

Del mismo Modo, existe Variables Globales que se declaran fuera del main de la siguiente manera:

**glb int** entero.,

**glb chr** char1,ch2,\_\_\_.,

**glb bln** booleano.,

# Main

Para que el programa pueda ser ejecutado es necesario que posea un main. Para poder especificar un main en Rill-20, se requiere hacer lo siguiente declaración:

()(**main**:

|Sentencias de la función|

:)

# Print

Imprime mensajes en consola.

(**pp**: "Hola Mundo!”:).,

Para agregar variables dentro de los mensajes que se quieren imprimir, se deben agregar con el símbolo “-” (menos) fuera de las comillas.

(**pp**: "La calificación es” - nota:).,

Para imprimir un salto de línea se utiliza la clausura “**ln**”.

(**pp**: **ln** :).,

(**pp**: "Hola Mundo!” - **ln** :).,

# Input

Para permitir el flujo de entrada estándar que normalmente es el teclado en tiempo de ejecución, se debe utilizar la siguiente sentencia.

(**in**: variable:).,

Que indica que el flujo de datos será asignado como valor a la variable.

# Comentarios

Rill-20 permite agregar comentarios al código fuente.

|Todo contenido entre estos símbolos serán considerados como comentario|

# Estructuras de Control

## **IF y SWITCH: una estructura unida**

Rill-20 combina las estructuras de control SWITCH-CASE e IF-ELSE de otros lenguajes, de tal manera que ambas tienen en mismo nombre de estructura de control “if”, puesto que su comportamiento es igual.

A pesar de que tengan el mismo nombre, dependiendo de la manera en que se declare, ésta se comportará ya sea como un SWITCH-CASE o como un IF-ELSE.

### IF Con Comportamiento IF-ELSE

Cuando la declaración de la función sea de la manera siguiente:

(condición)(**if**:

La estructura de control tendrá un comportamiento similar a la estructura de control IF-ELSE de otros lenguajes, y su estructura y comportamiento serán los siguientes:

(condición C1)(**if**:

|Sentencias que cumplen la condición C1|

(condición C2):

|Sentencias que cumplen la condición C2|

(**els**):

|Sentencias que no cumplen ninguna condición|

:)

#### Condiciones en IF Con Comportamiento IF-ELSE

Las condiciones dentro del esta estructura de control son de la siguiente forma:

(expresión1 **operador\_relacional** espresión2)

Los operadores relacionales que Rill-20 implementa se muestran en la tabla siguiente:

|  |  |  |  |
| --- | --- | --- | --- |
| Símbolo | Nombre | Uso | Ejemplo |
| = | Igual que |  |  |
| < | Menor que |  |  |
| > | Mayor que |  |  |
| <= | Menor que |  |  |
| => |  |  |  |
| != |  |  |  |

Además, se hace uso de las cláusulas **and** y **or** dentro de las condiciones.

#### Ejemplo de IF Con Comportamiento IF-ELSE

(puntaje >= 0 **and** puntaje < 59)(**if**:

|La nota es ‘R’ (Reprobado)|

nota: ‘R’ .,

(puntaje >= 60 **and** puntaje < 70):

|La nota es ‘D’ (Deficiente)|

nota: ‘D’ .,

(puntaje >= 70 **and** puntaje < 80):

|La nota es ‘C’ (Suficiente)|

nota: ‘C’ .,

(puntaje >= 80 **and** puntaje < 90):

|La nota es ‘B’ (Bien)|

nota: ‘B’ .,

(puntaje >= 90 **and** puntaje <= 100):

|La nota es ‘B’ (Excelente)|

nota: ‘A’ .,

(**els**):

|La nota es ‘E’ (Errónea)|

nota: ‘E’ .,

:)

### IF Con Comportamiento SWTCH-CASE

Cuando la declaración de la función sea de la manare siguiente:

(variable)(**if**:

La estructura de control tendrá un comportamiento similar a la estructura de control SWITCH-CASE de otros lenguajes, y su estructura y comportamiento serán los siguientes:

(variable)(**if**:

(case C1):

|Sentencias que cumplen el case C1|

(case C2):

|Sentencias que cumplen el case C2|

(**els**):

|Sentencias que no cumplen ninguna condición|

:)

#### Cases en IF Con Comportamiento SWTCH-CASE

Los Cases en esta estructura de control son de la siguiente manera:

(**operador\_relacional** variable)

Por ejemplo:

(**<** 0)

(**=** 0)

(1 **...** 100)

(> 100)

etc.

Donde la variable es la que se introdujo en la definición del IF Con Comportamiento SWTCH-CASE.

#### Clausura “…” en IF Con Comportamiento SWTCH-CASE

La estructura de IF Con Comportamiento SWTCH-CASE admite una clausura especial denotada por el símbolo “**...**” (puntos suspensivos) que indica que el valor de la variable de la estructura de control IF Con Comportamiento SWTCH-CASE está entre los dos valores dados.

Por ejemplo, el case (0 **...** 100)se cumple si la variable del IF con Comportamiento SWTCH-CASE tiene algún valor dentro de 0 y 100 incluyendo 0 y 100.

Nota: Si la más de un CASE es verdadero en la estructura de control, las sentencias de cada todos los CASES verdaderos serán ejecutadas.

#### Ejemplo de IF Con Comportamiento SWITCH-CASE

(puntaje)(**if**:

(0 **...** 59):

|La nota es ‘F’ (Reprobado)|

nota: ‘F’ .,

(60 **...** 69):

|La nota es ‘D’ (Deficiente)|

nota: ‘D’ .,

(70 **...** 79):

|La nota es ‘C’ (Suficiente)|

nota: ‘C’ .,

(80 **...** 90):

|La nota es ‘B’ (Bien)|

nota: ‘B’ .,

(90 **...** 100):

|La nota es ‘A’ (Excelente)|

nota: ‘A’ .,

(**els**):

|La nota no es válida ‘E’ (errónea)|

nota: ‘E’ .,

:)

## **FOR**

La estructura de control FOR posee la siguiente configuración:

(variable, 1 **to** 10, incremento)(**fr**:

:)

En esta estructura se tienen las siguientes consideraciones:

* Si la variable no está declarada en el ámbito exterior al bloque FOR (fr), ésta se considerará como una nueva variable, su tipo será INTEGER (int) y sólo serpa válida dentro del ámbito del bloque FOR.
* El incremento tiene la siguiente estructura:

+num

-num

Que indica de cuánto será el incremento de la variable que controla los ciclos del bloque FOR.

### Ejemplo de FOR

(a, 1 **to** 10, +1)(**fr**:

|Imprima “Hola Mundo!”|

(**pp**: "Hola Mundo!" + **ln** :).,

:)

## **WHILE**

La estructura de control WHILE tiene la siguiente estructura:

(condición)(**wh**:

|Sentencias que se ejecutan mientras la condición sea verdadera|

:)

### Ejemplo de WHILE

**int** a: 1 .,

(a <= 10)(**wh**:

|Imprima “Hola Mundo!”|

(**pp**: "Hola Mundo!" + **ln** :).,

:)

## **FUNCIONES Y PROCEDIMIENTOS**

Las funciones y procedimientos se diferencian en base a su tipo, al no tener tipo será una función y por lo tanto no retornará nada, mientras que al tener u tipo, será procedimiento y debe retornar algún valor.

Las configuraciones de ambas estructuras de control son las siguientes:

* Funciones

(**tipo** var1, **tipo** var2, ...) **NOMBRE\_FUNCIÓN** (**fun**: **nll**

|Sentencias de la función|

:)

* Procedimientos:

(**tipo** var1, **tipo** var2, ...) **NOMBRE\_PROCEDIMIENTO** (**fun**: **tipo**

|Sentencias del procedimiento|

**ret** valor .,

:)

### Ejemplos de Función y Procedimiento

(**int** puntaje) **getNota** (**fun**: **char**

(puntaje)(**if**:

(0 **...** 59):

|La nota es ‘F’ (Reprobado)|

**ret** ‘F’ .,

(60 **...** 69):

|La nota es ‘D’ (Deficiente)|

**ret** ‘D’ .,

(70 **...** 79):

|La nota es ‘C’ (Suficiente)|

**ret** ‘C’ .,

(80 **...** 90):

|La nota es ‘B’ (Bien)|

**ret** ‘B’ .,

(90 **...** 100):

|La nota es ‘A’ (Excelente)|

**ret** ‘A’ .,

(**els**):

|La nota no es válida ‘E’ (errónea)|

**ret** ‘E’ .,

:)

:)

# Arreglos Unidimensionales y Bidimensionales

Para la declaración de la arreglos Unidimensionales se utiliza la siguiente sentencia:

|Declaración de un arreglo vacío|

**mtx[**tamaño**]** **int** array .,

|Declaración de un arreglo con elementos|

**mtx** **int** array: [1,2,3,4] .,

Para la declaración de la arreglos Unidimensionales se utiliza la siguiente sentencia:

|Declaración de un arreglo vacío|

**mtx[**filas**][**columnas**]** **int** matrix .,

|Declaración de un arreglo con elementos|

**mtx** **int** matrix: [[1,2,3,4], [1,2,3,4], [1,2,3,4], [1,2,3,4]] .,

Todos los elementos dentro de arreglo o de la matriz deben ser del mismo tipo.

# Manual de Usuario

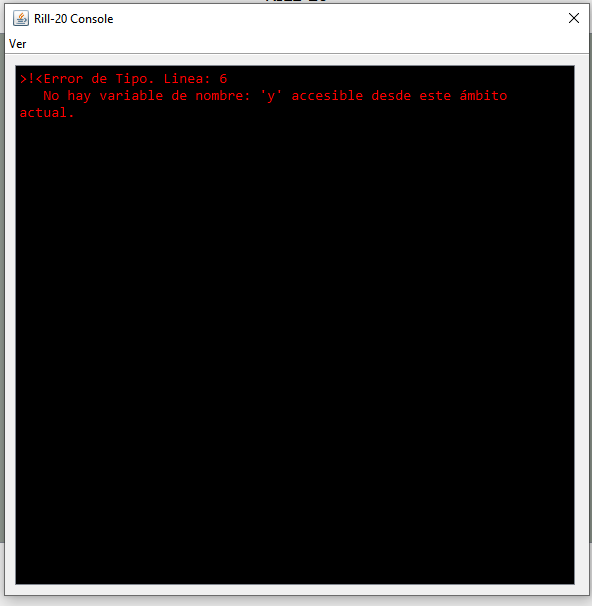
El programa editor de texto consiste en una ventana que posee una única ara de texto que en la que se escribe el código:



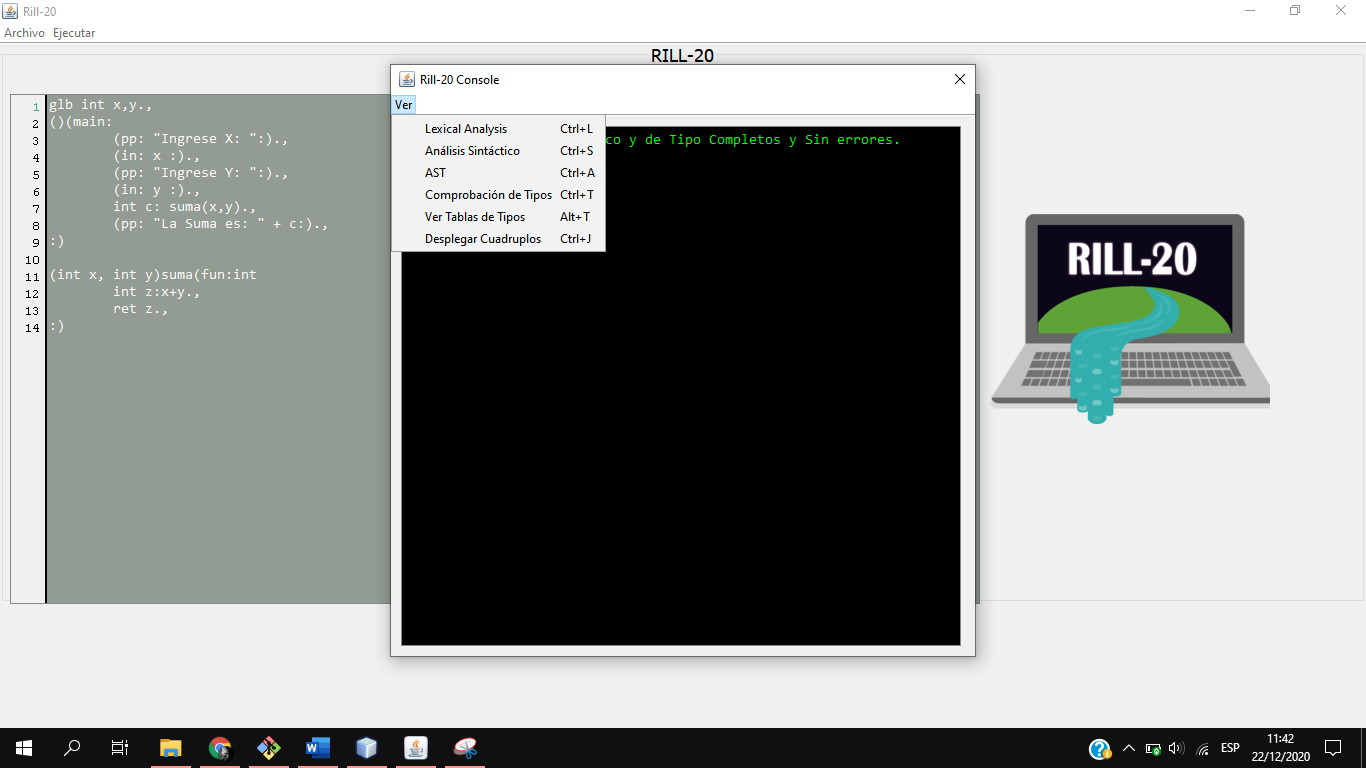
Del mismo modo se pueden importar archivos dando clic en el ítem de menú **Archivo.**

Para ejecutar el código del área de texto se puede usar ya sea la tecla **F9** o dando clic en el ítem de menú **Ejecutar**

Al ejecutar el archivo se mostrará una ventana donde se muestran los errores que el código tiene y en qué línea dichos errores están ocurriendo.

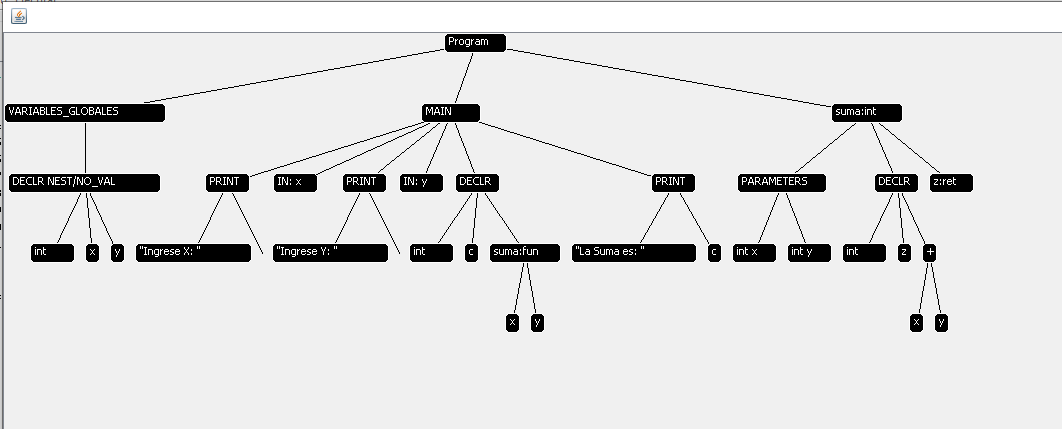


Al no tener errores el código, se ejecutarán todas las fases de compilación y se guardará un archivo ejecutable en SPIM20 que tiene el código máquina generado por el código escrito en el área de texto.

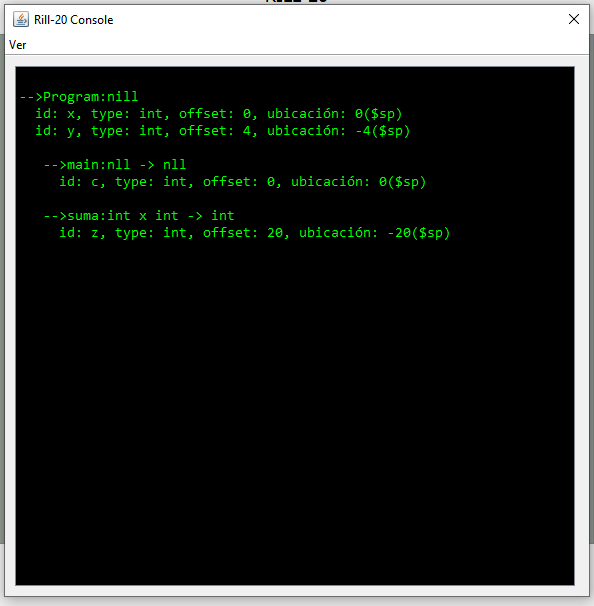


También se puede ejecutar cada fase de compilación individualmente, y se tienen opciones como ser:

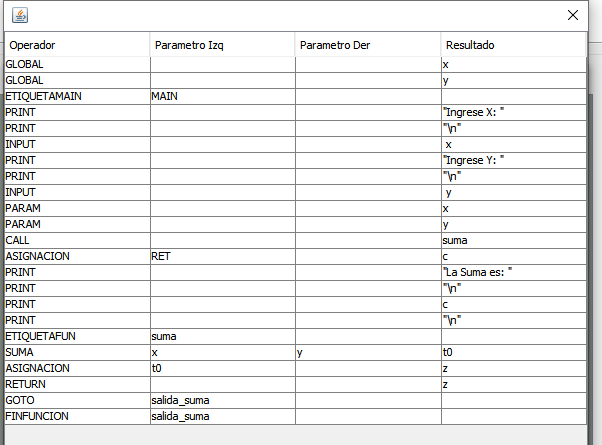
* AST: Que muestra el AST generado por el código



* Ver tabla de Tipos: que muestra las tablas de tipo de cada función y estructura de control.



* Despliegue cuádruplos: que muestra el código intermedio generado.



#### Ejemplo de Código 1

glb int x,y.,

()(main:

(pp: "Ingrese X: ":).,

(in: x :).,

(pp: "Ingrese Y: ":).,

(in: y :).,

int c: suma(x,y).,

(pp: "La Suma es: " + c:).,

:)

(int x, int y)suma(fun:int

int z:x+y.,

ret z.,

:)

#### Ejemplo de Código 2

()(main:

int a.,

(pp: "Ingrese un numero: " :).,

(in: a :).,

int b: fib(a).,

(pp: "Resultado: "+b:).,

:)

(int n)fib(fun:int

int x.,

(n = 1)(if:

x: 1.,

:)

(n = 2)(if:

x: 1.,

:)

int k: n - 1.,

int p: n - 2.,

int fiba: fib(k).,

int fibb: fib(p).,

x: fiba + fibb.,

ret x.,

:)

#### Ejemplo de Código 3

()(main:

chr nota., |Declaración de Variables|

int puntaje:-1., |Declaración de Variables|

(pp: "Ingrese un puntaje: ":)., |Print|

(in: puntaje:).,|Cin|

(puntaje < 0 or puntaje>100 )(wh:

(pp: "Nota no válida. Ingrese un puntaje: ":).,

(in: puntaje:).,

:)| FIN WHILE|

(puntaje)(if:

(0 ... 59):

nota: 'F' ., |La nota es ‘F’ (Reprobado)|

(pp: "La nota es" - nota - "(Reprobado)" - ln:).,

(60 ... 69):

nota: 'D' ., |La nota es ‘D’ (Deficiente)|

(pp: "La nota es" - nota - "(Deficiente)" - ln:).,

(70 ... 79):

nota: 'C' ., |La nota es ‘C’ (Suficiente)|

(pp: "La nota es" - nota - "(Suficiente)" - ln:).,

(80 ... 90):

nota: 'B' ., |La nota es ‘B’ (Bien)|

(pp: "La nota es" - nota - "(Bien)" - ln:).,

(90 ... 100):

nota: 'A' ., |La nota es ‘A’ (Excelente)|

(pp: "La nota es" - nota - "(Excelente)" - ln:).,

(els):

nota: 'E' ., |La nota no es válida ‘E’ (errónea)|

(pp: "La nota es" - nota - "(errónea)" - ln:).,

:)|SWITCH|

:)