Presentación de RAPTOR

RAPTOR es el acrónimo de :

Rapid Algorithmic Prototyping Tool for Ordered Reasoning

Que traducido quiere decir:
Herramienta de creación **R**ápida de **P**rototipos **A**lgorítmicos para **R**azonamiento **O**rdenado

Descargar desde la Plataforma Moodle
o desde:
http://raptor.martincarlisle.com/

Try now

Download RAPTOR using button below

Where/how are you using RAPTOR? I keep a list of what schools and universities are using RAPTOR and for what class. This helps me plan future advancements. Please email me and let me know.

Newest Installer (11/19/2019)

Download latest version

Descarga última version instalable

DOWNLOAD RAPTOR HERE!!

NEW FEATURES:

- To_Integer() takes in a string and returns the integer number (e.g. To_Integer("37") is 37)
- To_Float() takes in a string and returns the number (e.g. To_Float("37.5") is 37.5)
- Get_Nth_String returns the nth string in a long string, separated by the separator (e.g. Get_Nth_String("ab,cd,ef",",",2) is "cd"). It returns the separator after the last separated string (e.g. Get_Nth_String("ab,cd,ef",",",4) returns ",")

Digitally Signed Installer (older version, digitally signed on 10/1/2016 -- may help if you have Windows Defender issues!)

Digitally Signed Installer

Based on .NET Framework 4.5. XP users may need to use an older installer (2014 or earlier).

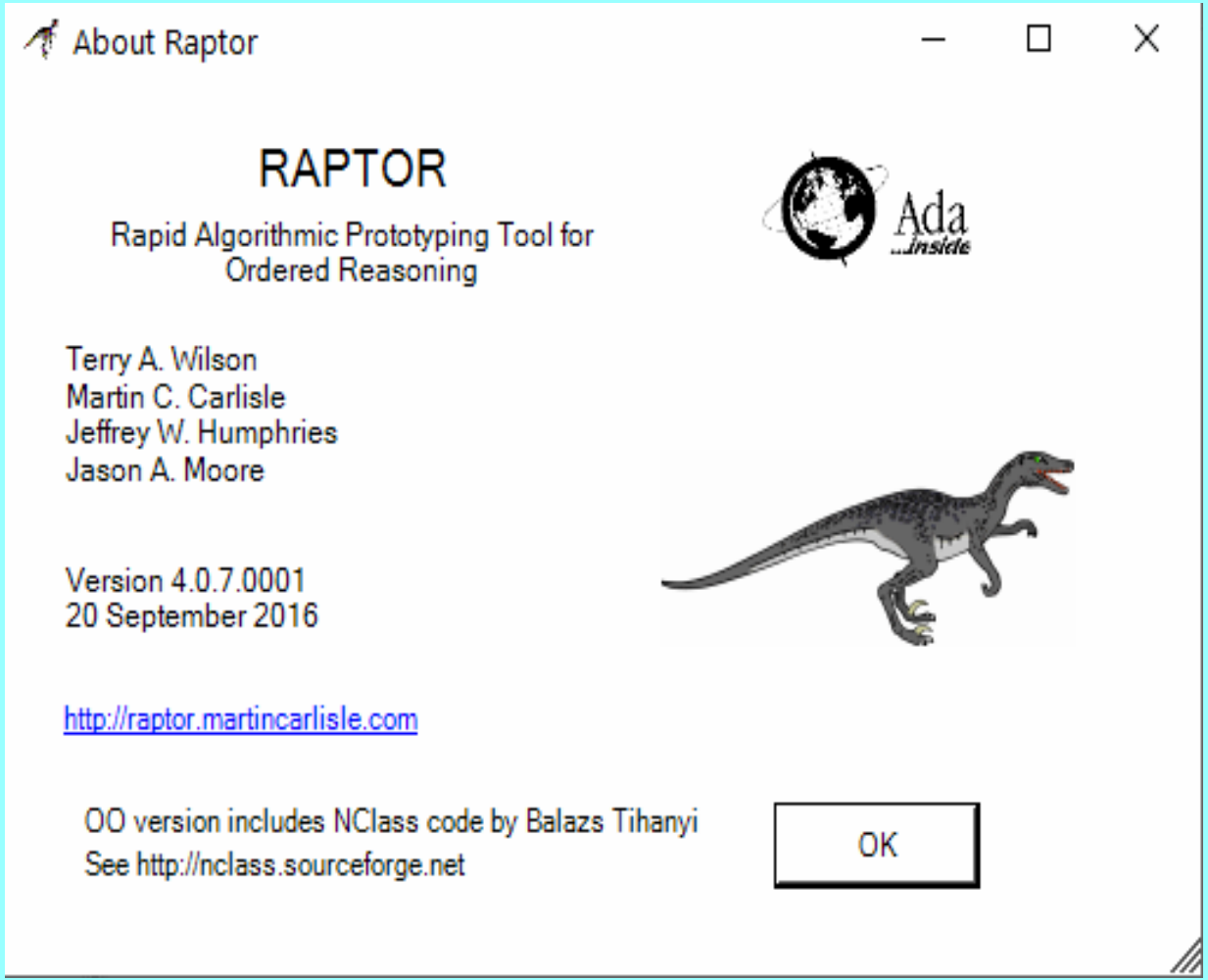
Descarga última version Portable para Instalar en USB

Portable Version

John Meir from Midlands Tech created a PortableApp version (PortableApps.com). This allows RAPTOR to be used from a USB key or similar without installing. You can download the portable version here. This version is from 2012.

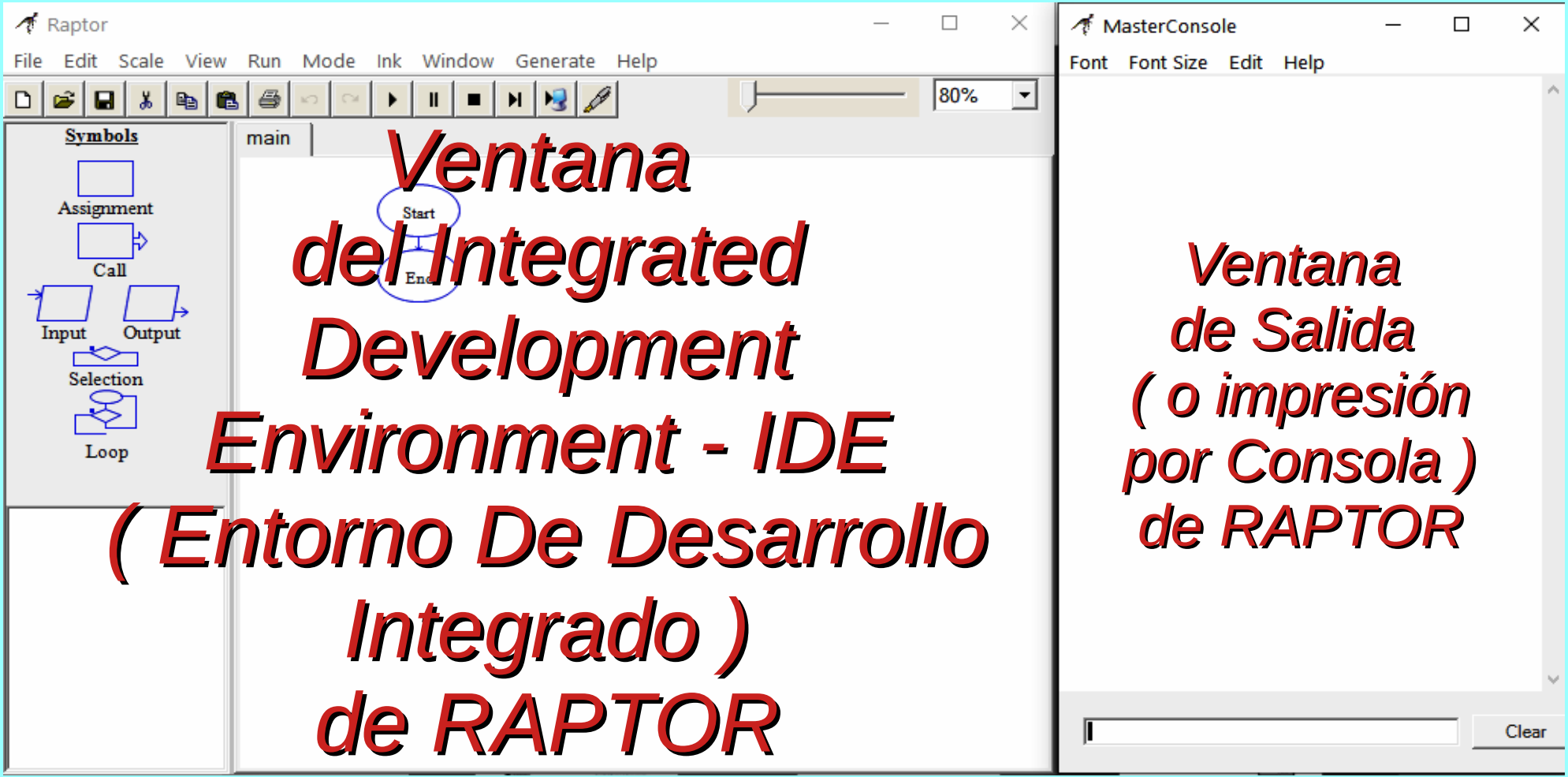
FALL 2015 VERSION (Updated 15 August 2015)

Ventana sobre RAPTOR

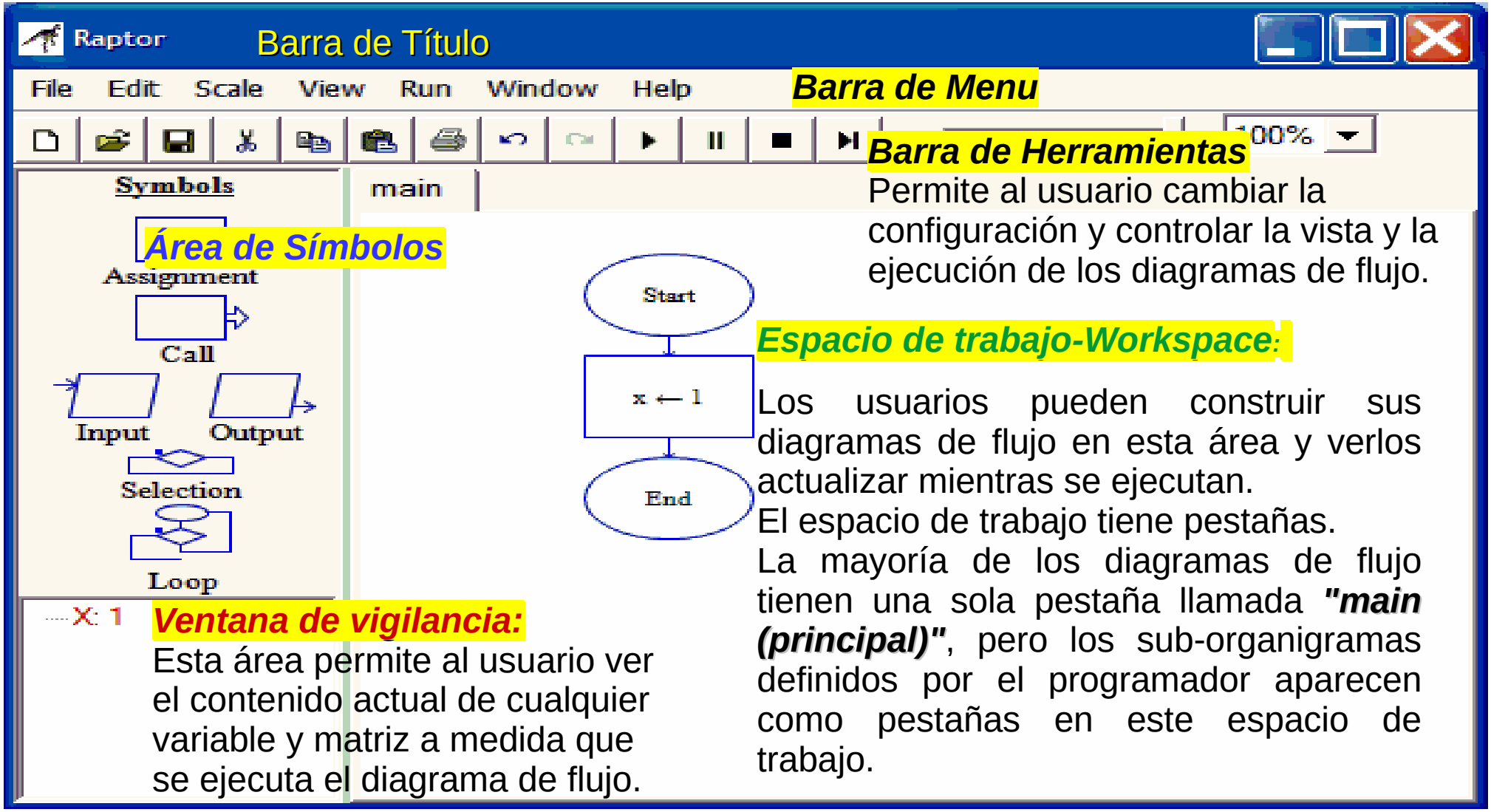


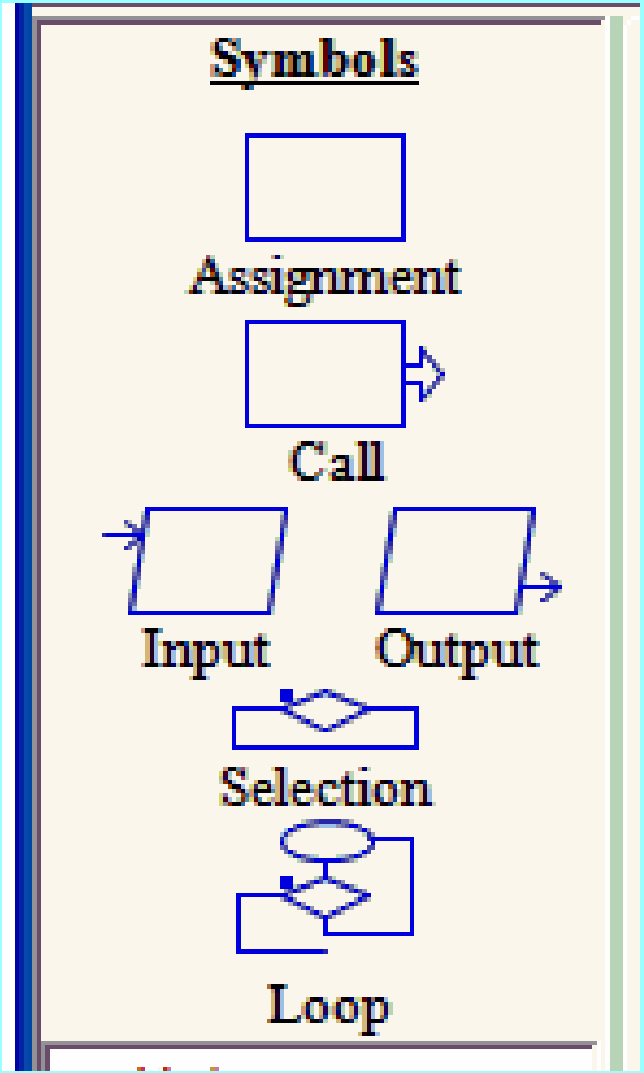
Raptor es una herramienta de resolución de problemas fácil de usar que permite al usuario ***generar diagramas de flujo ejecutables.*** Raptor fue escrito para estudiantes que están siendo introducidos en la disciplina informática para desarrollar habilidades de resolución de problemas y mejorar el pensamiento ***algorítmico***.

Ventanas de la Interfaz de RAPTOR



La ventana principal consta de cuatro áreas:





El ***área de Símbolos*** presenta los 6 símbolos gráficos principales que se pueden usar en Raptor

- El ***símbolo de asignación-Assignment*** se usa para dar a una variable un valor numérico o de texto.
- El ***símbolo de llamada-Call*** se utiliza para realizar llamadas a procedimientos externos, como rutinas de gráficos.
- El ***símbolo de entrada-Input*** se utiliza para obtener información del usuario.
- El ***símbolo de salida-Output*** se utiliza para mostrar texto en la **consola** maestra.
- La ***estructura de selección-Selection*** se utiliza para la toma de decisiones.
- La ***estructura de bucle-Loop*** se usa para iteración y repetición.

Nombres: reglas y sugerencias

Debido a que Raptor es un lenguaje gráfico, los programas Raptor tienden a depender menos de las palabras que la mayoría de los lenguajes basados en texto. Sin embargo, las palabras todavía aparecen en Raptor como nombres de variables, nombres de sub-charts, nombres de procedimientos y nombres de funciones. En los círculos informáticos, estos nombres se conocen como **identificadores**, y los programas deben seguir ciertas reglas para crear nombres o **identificadores** correctos.

Reglas para nombres en Raptor *(Ver: 3 - Guia_U3_TIPOSdeDATOS PeC)*

- Un ***nombre*** debe comenzar con una ***letra***
- ***Después de la primera letra puede aparecer*** cualquier letra, dígitos o guiones bajos
- ***No se permiten espacios*** en un nombre: un espacio marca el final de un nombre.
- Los nombres NO distinguen entre mayúsculas y minúsculas (en RAPTOR). Por lo tanto,
  - **get_mouse_button** es lo mismo que **Get_Mouse_Button** y **GET_MOUSE_BUTTON**
  - **Contar** significa lo mismo que **contar**
- ***Un nombre NO puede representar más de una cosa*** en un programa.
  - Por lo tanto un programa no puede usar una variable llamada ***e,*** porque ***e*** está predefinida en Raptor; no puede usar la variable llamada ***red***, porque ***red*** está predefinido; un programador no puede crear un sub-chart llamado **Get_Key**, porque **Get_Key** ya está definido
- ***Además de las reglas anteriores que Raptor aplica sin excepción, existen convenciones de nomenclatura*** comunes que deben usarse para evitar confusiones, hacer que los programas sean legibles y minimizar la ocurrencia de errores.

Sugerencias para nombres de variables significativas en Raptor

Los programadores usan nombres de variables significativos para que los propios nombres identifiquen para qué se usa cada variable.

A continuación se presentan algunas reglas generales para nombrar variables:

- Utilice nombres de variables compuestos de palabras que describan el contenido de la variable.
  - **velocidad** es un nombre mejor que **v**
- Use **guiones bajos** para que las palabras compuestas sean legibles
  - **primera_velocidad** es un nombre mejor que **pv** o **primeravelocidad**
- Al realizar conversiones entre unidades, haga que las **unidades** formen parte del nombre de la variable
- **Beta_radian** y **Beta_grados** dejan pocas dudas sobre las unidades para los valores almacenados

Variables en Raptor

Una variable puede considerarse como un **nombre** asociado con un **valor**. A una variable se le asigna un valor en un **símbolo de entrada** o **en un símbolo de asignación.** La primera vez que se le da un valor a una variable, se establecen ciertos atributos para la variable:

- Su **tipo**: ¿contiene una cadena o un número?
- Su **estructura**: ¿tiene un valor único o es parte de una matriz?

Una vez que se establecen estos atributos, **NO** pueden cambiar. El nombre siempre estará asociado con el **tipo** y **estructura** iniciales. Una variable inicializada como una cadena siempre debe usarse como una cadena. Una variable inicializada como una matriz unidimensional de valores numéricos no se puede cambiar a una variable escalar (única) más adelante. Un **programa** puede determinar el **tipo** y la **estructura** de una variable utilizando las **funciones**: Is_String, Is_Array, Is_Number, Is_2D_Array.

Tipo de una variable

El **tipo** de una variable se establece cuando la variable recibe un valor por primera vez. El **tipo** determina las **operaciones** que se pueden realizar en la variable.
Para las variables **numéricas**, esas **operaciones** se entienden bastante bien (consulte *Math Operators, Math (Non-Trigonometric) Functions, y Trigonometric Functions*). Las **operaciones** en **cadenas (string)** son menos comprendidas (ver *String Operations).* Las **comparaciones numéricas** también son familiares (ver Boolean Expressions); ***comparaciones similares están disponibles para cadenas*** (ver *String Comparisons).*Si los programas tratan solo con datos numéricos, surgen pocos problemas con el tipo de datos. Al mezclar el uso de cadenas y números, consulte *String Variables & Assignment y String vs. Numeric Input* para obtener una comprensión completa de cómo se establecen inicialmente los tipos.

Estructura de una variable: matrices

Las matrices proporcionan a los programas la capacidad de tratar en forma simple grandes cantidades de datos (consulte *Array Overview*). Con las matrices, un nombre representa muchas variables (y sus valores respectivos, todos del mismo tipo). Raptor admite matrices unidimensionales y bidimensionales.

El espaciado mejora la legibilidad

Debido a que Raptor es un lenguaje gráfico, los programas Raptor tienden a depender menos de las palabras que la mayoría de los lenguajes basados en texto. Sin embargo, cómo se escriben los programas puede afectar nuestra comprensión de lo que hacen. En particular, el uso cuidadoso del espacio puede hacer que los programas sean más legibles.

Espaciado en cuadros de texto Raptor

Como se señaló en Nombres: reglas y sugerencias, no se permiten espacios en el medio de un nombre. Sin embargo, se permiten espacios entre diferentes elementos que el usuario escribe en un cuadro de texto. A menudo, agregar espacios puede hacer que el programa sea más legible.

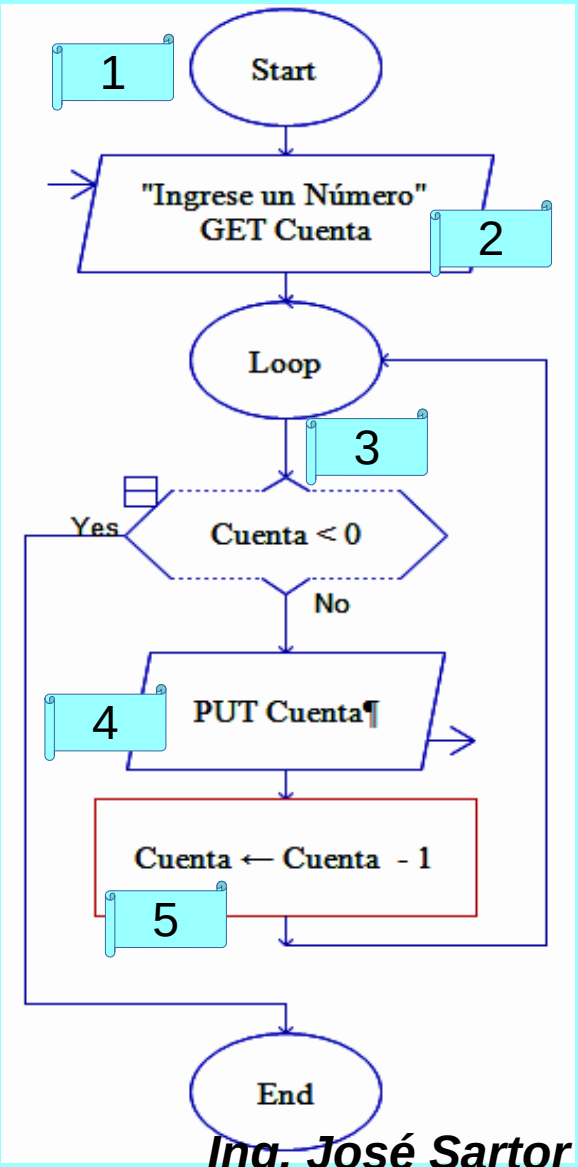
Por ejemplo, se pueden agregar espacios entre argumentos en una llamada a procedimiento:

Display_Text (100,100,"Puntuación:"+Puntuación,Azul) vs.
Display_Text (100, 100, "Puntuación:" + Puntuación, Azul)

En el rombo de una selección: Cuenta=5 vs. ***cuenta = 5***

En una expresión matemática:
Fahrenheit=Celcius*9/5 +32 vs. ***Fahrenheit = (Celcius * (9 / 5)) + 32***

En todos los casos anteriores, la segunda versión es mucho más legible.



Tutorial - Crear un diagrama de flujo (Flowchart)

Este breve tutorial lo guiará a través de la construcción y ejecución de un diagrama de flujo simple, que le pida un número al usuario y luego haga una cuenta regresiva hasta cero (mostrando cada número en la Consola).

1. **Inicie Raptor.** Verá el *Inicio-Start* y *Fin-End* en el área gráfica.

2. Agregue un **símbolo de Entrada** al diagrama de flujo *haciendo clic* izquierdo una vez en *el símbolo de Entrada* y luego *agregándolo entre Inicio y Fin*. Haga doble clic para agregar el mensaje ***"Ingrese un número:"*** y use el nombre de la variable ***Cuenta***.

3. A continuación, agregue una **estructura de bucle** y edite su expresión a ***Cuenta < 0***

4. Agregue un **símbolo de Salida** para mostrar la variable **Cuenta**.

5. Agregue un **símbolo de Asignación** para disminuir el valor de **Cuenta**. Edite el contenido a: ***Cuenta = Cuenta – 1***

6. Para ejecutar el diagrama de flujo, haga clic en: