Algoritmos y Estructura de Datos

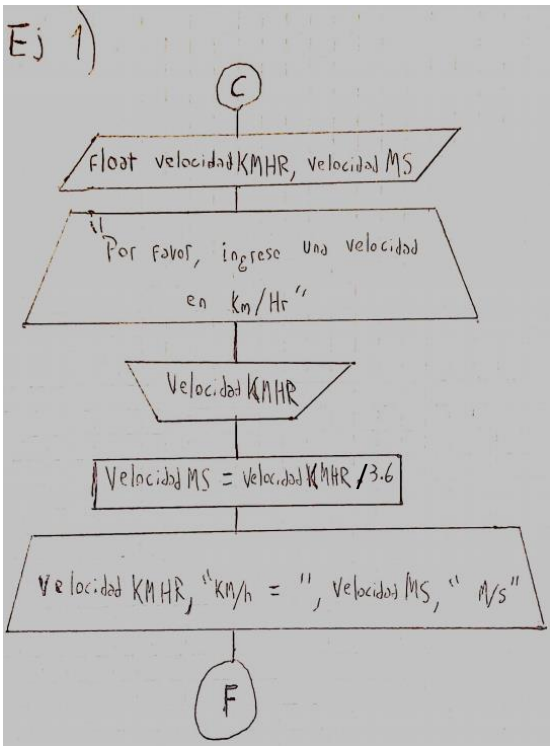
Ing. Pablo D. Mendez

“Sentencias de Asignación y Sentencias Selectivas”

Legajo: 2047792
Nombre y Apellido: Pedro Baccaro
Mail Institucional: pbaccaro@frba.utn.edu.ar
Usuario de GitHub: pedro-baccaro
Link al Repositorio: https://github.com/pedro-baccaro/AED-2021

Ejercicio 1

Diagrama de Lindsay



Análisis del Problema

El objetivo del programa es permitirle al usuario convertir velocidades en kilómetros por hora a velocidades en metros por segundo. Esto se logra siguiendo una serie de pasos:

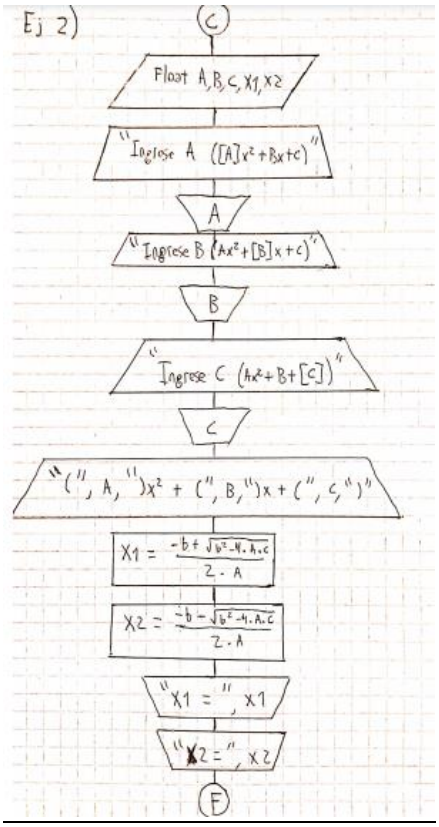
En primer lugar, el programa solicita al usuario el valor de una velocidad en kilómetros por hora. Este valor puede ser positivo o negativo, y de prácticamente cualquier magnitud (se aclara prácticamente porque las limitaciones del programa y la computadora hacen que valores muy altos o bajos no devuelvan resultados acertados).

Luego de obtener este valor, que representa una velocidad en kilómetros por hora, el programa se encarga de convertirla de km/h a m/s. Esto lo realiza dividiendo el valor ingresado por 3,6.
¿Por qué 3,6? Es simple, un kilómetro es igual a 1000 metros, mientras que una hora es igual a 3600 segundos. Como estamos pasando de kilómetro a metro, y de hora a segundo, debemos dividir el valor por 1000, y luego dividirlo por 3600. Para ahorrarnos un paso, dividimos al 3600 por 1000 y nos quedamos con 3,6.

Por último, el programa le muestra al usuario a cuantos m/s equivale el valor km/h que ingresó.

Ejercicio 2

Diagrama de Lindsay



Análisis del Problema

El objetivo de este programa es resolver lo que se conoce como una ecuación cuadrática. Esta se caracteriza por tener la forma $Ax^2+Bx+C = 0$, siendo A, B, C y X valores numéricos. En este tipo de ecuación se conocen A, B y C y se quiere saber los valores de X que hacen que la ecuación sea igual a 0, también conocidos como las raíces de la ecuación.

En primer lugar, el programa pide al usuario que ingrese los valores de A, B, y C, para conocer la ecuación que se busca resolver. Además, muestra en que parte de la ecuación se ubica dicho valor para facilitar el entendimiento (lo indica entre corchetes).

En segundo lugar, se utiliza lo que se conoce como fórmula resolvente. Esta tiene la forma $\frac{-b\pm\sqrt{b^2-4\times a\times c}}{2\times a}$ y puede tener una, dos, o cero soluciones (esto sucede por el "±" en la ecuación, que genera que haya un resultado si se usa "+", y otro si se usa "-").

Dependiendo de la cantidad de soluciones (o raíces) que posee la ecuación, el programa actuará tres posibles maneras:

- *Cero soluciones*

  Devolverá 2 veces el valor “nan” (que significa “not a number”). Esto sucede porque en algún lado de la ecuación se intentó calcular una división por 0 o la raíz cuadrada de un número negativo. Estas dos operaciones no tienen resultado (salvo que usemos números complejos), por lo que el programa asigna el valor nan a las raíces.

  Ejemplo:

```
//Input:
3x²+5x+7

//Output:
X1 = nan
X2 = nan
```

- *Una solución*

  Devolverá 2 veces un mismo valor. Esto sucede porque las ecuaciones cuadráticas tienen una sola solución cuando ambas raíces son iguales. Esto se conoce como “raíz doble”.

  Ejemplo:

```
//Input:
1x²-4x+4

//Output:
X1 = 3
X2 = 3
```

- *Dos soluciones*

  Devolverá ambas soluciones. Esto significa que cualquiera de los 2 valores de x hará que la ecuación de 0 como resultado.

  Ejemplo:

```
//Input:
1x²-5x+6

//Output:
X1 = 2
X2 = 5
```

Ejercicio 3:

Variables en JavaScript

Este lenguaje se diferencia de C++, entre otras cosas, por su forma de declarar variables. En vez de declarar el tipo de dato que se guardará en la variable, simplemente se declara su nombre y un valor inicial, y se permite más adelante reasignarle un valor (aunque este sea otro tipo de valor).

Por ejemplo, mientras que en C++ uno escribiría int X = 5 y declararía que este será un entero durante toda la duración del programa, en JavaScript podría escribir var X = 5 y más adelante reasignar X escribiendo X = "Hola Mundo".

JavaScript es lo que se conoce como un lenguaje **débilmente tipado**, mientras que C++ es un lenguaje **fuertemente tipado**.

El tipado débil tiene sus ventajas y desventajas, ya que mientras genera en ocasiones más versatilidad, también puede producir errores si no prestamos atención. Por ejemplo:

```
A = 5
B = "hola"

Z = A+B
```

En vez de producirse un error aquí, Z es asignada el valor "5hola" si usamos JavaScript. Esto puede ser un problema ya que podemos no enterarnos de que el tipo de dato con el que estamos trabajando en un determinado momento.

Igualmente, al hablar de C++, un lenguaje fuertemente tipado, debemos notar que, aunque es más expresivo (porque *expresa* que tipo de dato se usará), eso lo vuelve más rígido en algunos aspectos. Por ejemplo:

Si quisiera que un usuario elija entre las opciones hola, adiós, 34, true no podría cambiar el tipo de la variable en la que se almacenará su respuesta, por lo que debería usar strings para todas ellas.

Por último, las variables en JavaScript pueden ser declaradas con los prefijos var y let (o sin prefijo, que es lo mismo que var). Esto determina el *alcance* de la variable, es decir, si su asignación es aplicada exclusivamente dentro de una función/bloque o *"globalmente"*. Una variable 'var' es global, mientras que una variable 'let' es local. Además, se puede utilizar 'const' para variables que no pueden ser reasignadas, que son constantes.

Sentencias If en JavaScript

Estas son bastante similares en ambos lenguajes en cuanto a su sintaxis, solamente las condiciones se diferencian en algunos aspectos gracias a la diferencia entre el tipo de tipado que ambos lenguajes utilizan. La lógica en JavaScript permite comparaciones entre diferentes tipos, lo que a veces da resultados inesperados. Aquí un par de ejemplos de esto:

```
X = 5

// == (es igual a)
X == 8          //falso
X == 5          //verdadero
X == "5"        //verdadero

// === (es igual a y tiene el mismo tipo que)
X === 5         //verdadero
X === "5"       //falso

// === (no es igual a)
X != 8          //verdadero
X != "5"        //falso

// !== (no es igual a o tiene distinto tipo que)
X !== 5         //falso
X !== "5"       //verdadero
```

Como podemos ver, al comparar diferentes tipos de datos en JS estos son a veces comparados como si fueran del mismo. Esto es al mismo tiempo útil, ya que permite hacer crear menos variables nuevas para cambiar un valor de tipo, y confuso para nuevos programadores, ya que a veces los resultados son inesperados.

Otra de las diferencias, por ejemplo, es la comparación entre 2 strings. Si en C++ busco saber el valor de verdad de "hola" > "casa" obtendría un error porque estos no pueden ser comparados. En JavaScript, sin embargo, la comparación "hola" > "casa" devuelve true, ya que los ordena alfabéticamente y luego compara cual es "mayor" en esa comparación.

Sean o no sean intuitivas las comparaciones que JavaScript permite, es interesante leer sobre estas para ver como lenguajes de programación diferentes a los que uno conoce funcionan.