Trabajo Práctico Nro 1
Algoritmos y estructura de datos
K1025

Tema: Sentencias de asignación y sentencias selectivas.

Fecha de entrega: 5/5

Nombre y apellido: Milagros Trejo
Legajo: 2046118
Correo institucional: mitrejo@frba.utn.edu.ar
Usuario GitHub: Milagros-Trejo
Link repositorio: https://github.com/Milagros-Trejo/AyED_K1025_2021_Trejo

Consigna 3.a)

En este ejercicio nos piden armar un conversor de km/h a m/s, dos unidades de velocidad.
Primero hay que saber qué operación u operaciones son necesarias para el pase.
Tengamos en claro que:

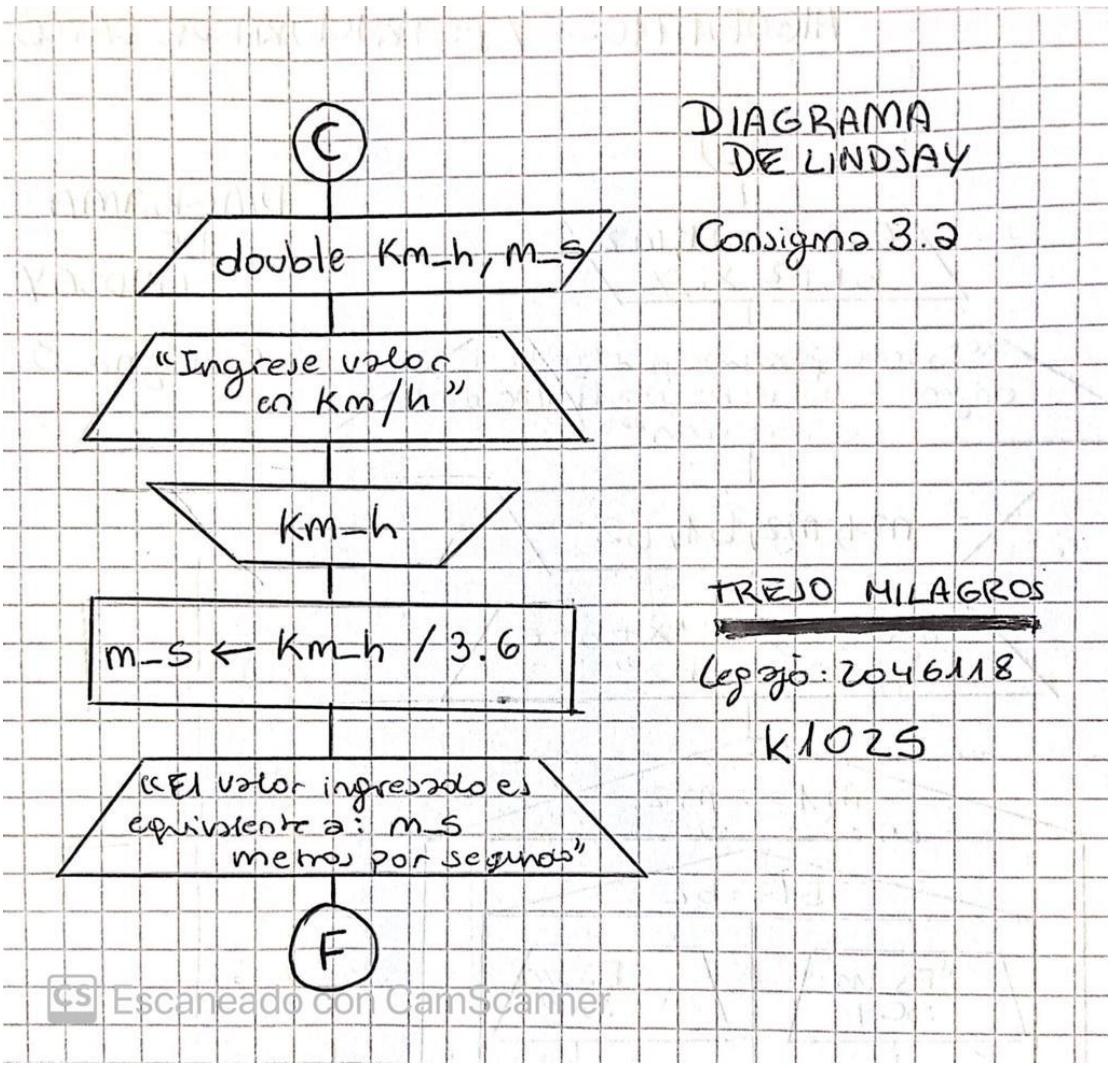
$1km = 1000m$ y $1h = 3600\ seg$

Eso hace que la conversión tenga la forma

$$\frac{1\ km}{1h} \cdot \frac{1000m}{1km} \cdot \frac{1h}{3600\ seg} = \frac{1}{1} \cdot \frac{1000m}{3600\ seg}$$

(que simplificando, es equivalente a dividir los km/h por 3,6)

Una vez establecida la operación de conversión, solo hay que pedirle al usuario que ingrese valores en km/h para devolver un valor en m/s después de haber operado, como se muestra en el diagrama de Lindsay.



Consigna 3.b)

En este ejercicio se nos pide encontrar la intersección de dos rectas ingresadas por el usuario. Para empezar a resolver este problema, hay que primero hacer el análisis matemático correspondiente para saber cómo operar con las variables ingresadas:
Llamemos por conveniencia “m” a la primera pendiente, “n” a la segunda; “b” a la primera ordenada al origen, y “c” a la segunda.

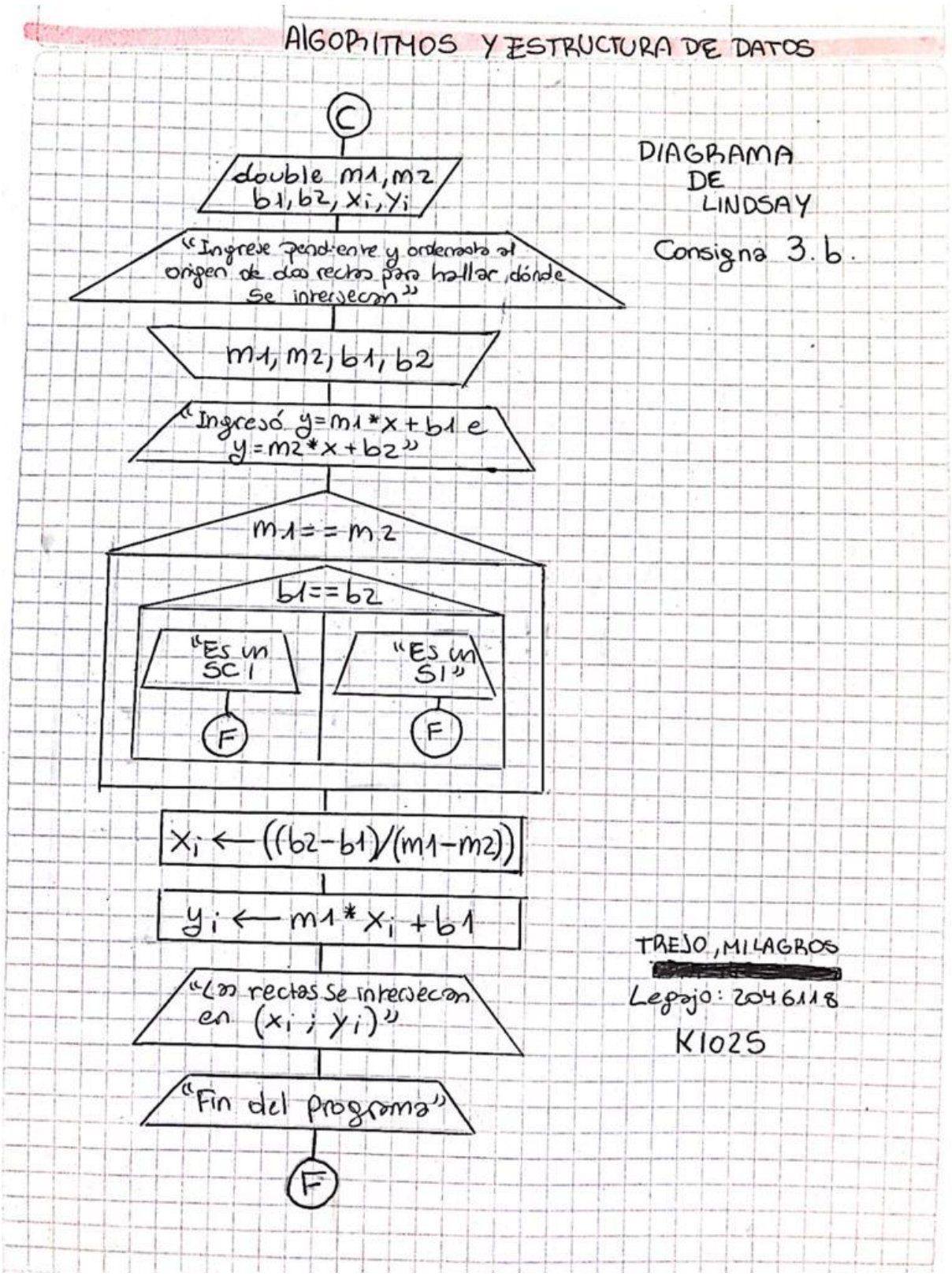
$$y = mx + b$$
$$y = nx + c$$

Sabemos que para resolver un sistema de este tipo debemos igualar las ecuaciones suponiendo que y es igual en ambos casos (ya que es la coordenada y de intersección).

$$mx + b = nx + c$$
$$mx - nx = c - b$$

$$(m - n)x = c - b$$
$$x = \frac{c-b}{m-n}$$

Eso ya nos permite averiguar la x para cualquier valor de m,n,b,c en caso de que sea un sistema compatible determinado (está claro que m debe ser distinto de n para no dividir por cero, pero ya lo trataré más adelante). Una vez que obtengamos la coordenada *x* de la intersección, se puede fácilmente reemplazar en cualquiera de las dos ecuaciones de las rectas para hallar la coordenada *y* del punto intersección.
Sin embargo, en un sistema de dos rectas, se puede dar el caso de que sea compatible indeterminado o incluso incompatible, por lo que antes de empezar a operar con las variables hay que asegurarnos de que no estamos en presencia de alguna de estas situaciones.
En caso de que sea un sistema compatible indeterminado, sabemos por definición que son la misma recta, por lo tanto si m=n y al mismo tiempo b=c, estamos en presencia de dos rectas que en realidad son iguales. Si esas dos condiciones se cumplen simultáneamente, es un sistema compatible indeterminado (SCI).
En caso de que m=n y b no sea igual que c, nos encontramos frente a un sistema incompatible (SI), ya que las pendientes de ambas rectas son iguales, pero tienen distinta ordenada a origen. Esto se traduce en dos rectas paralelas que no se intersecan.
En el diagrama a continuación y en el código se detallan estas condiciones.



(A efectos de que el diagrama sea más prolijo, cambian algunos textos del cout en el código aunque aquí estén presentados de otra manera)

Como se ve, el procedimiento consiste en pedir al usuario que ingrese valores para las cuatro variables, comprobar que no sea un sistema compatible indeterminado o

incompatible, y recién en este punto operar para encontrar la intersección. Una vez que obtuvimos las coordenadas *x* e *y* del punto intersección, utilizamos un cout para mostrarlo al usuario.

Consigna 4: Declaración de variables y sentencias "if" en Javascript vs C/C++

Declaración de variables

La primera diferencia que salta a la vista entre la declaración de variables en Javascript (JS) y la de C++, es que C++ es un lenguaje fuertemente tipado y JS no. Un lenguaje fuertemente tipado es aquel que trabaja con variables con un tipo de dato definido, que no se puede cambiar (salvo que se haga una conversión). En C/C++, se define el tipo de dato a la hora de definir una variable, por ejemplo:

```
int y; // se declara como integer
double x; // se declara como double
```

Cuando estamos en JS esto no sucede, la variable se declara sin el tipo de dato y se le puede asignar cualquier tipo de variable, por lo que se llama lenguaje débilmente tipado. A continuación veremos cómo se definen entonces las variables.

En Javascript hay tres formas de declarar una variable[1]:

La primera es con la sentencia "var", que simplemente enuncia la variable y le puede asignar opcionalmente un valor, como variable global o local. Para declarar dos variables debe separarse por punto y coma. Por ejemplo:

```
var y; // define la variable sin un valor asociado
    var x = 12; // asigna un valor al declarar
```

A diferencia del C/C++, las variables en JS pueden declararse con "var" luego de ser referenciadas. Se puede utilizar la variable para una sentencia y que la declaración esté más adelante en el código, pero no es una práctica recomendable.

La segunda manera es con "let", que funciona igual que var pero dentro de un ámbito de bloque (es decir, solo la declara dentro del bloque donde está trabajando, por ejemplo un bucle o un if, y solo servirá para ser referenciada dentro de ese bloque):

```
let f; let x = 34;
```

Una similitud con C/C++ es que las variables declaradas pueden ser globales, locales, o en ámbito de bloque. Las locales actúan dentro de la función donde fueron declaradas, las globales pueden referenciarse desde dentro o fuera de una

[1] Fuente: developer.mozilla.org

función, y las de ámbito de bloque funcionan en su bloque. Sin embargo, en C/C++ no se declaran de forma diferente.

Similarmente al let se puede usar el “const” para declarar una variable en ámbito de bloque, aunque como su nombre indica, es una constante de solo lectura. Esto significa que no pueden redefinirse más adelante.

```
const x = 'hola';
```

Finalmente, la tercera y última forma no es recomendable, y es simplemente no declararla. JS permite asignar valores a una variable que no ha sido declarada, y la crea como variable global no declarada. Esto, aunque posible, genera advertencias y puede traer problemas dependiendo de la situación, por lo que no es un buen uso del lenguaje.

A modo de sumario, un pequeño cuadro que muestra las principales diferencias y similitudes de JS con C/C++ en la declaración de variables:

C/C++	**Javascript**
Es un lenguaje fuertemente tipado	Es un lenguaje débilmente tipado
Se pueden declarar varias variables del mismo tipo en una línea (Ej: `int a,b;`)	Se debe separar por “;” cada vez que se quiera declarar una variable en la misma línea
Deben declararse las variables antes de ser referenciadas en cualquier caso	En algunos casos, una variable global puede referenciarse antes de ser declarada en el código, y en ocasiones ni siquiera es necesario declararla (aunque no sea recomendable)
Las variables pueden ser globales, locales o en ámbito de bloque	
Las variables en ámbito de bloque se declaran igual que las globales y locales	Hay formas especiales para declarar una variable de ámbito de bloque
Se debe asignar un identificador a la variable cuando se la declara	

Sentencias “if”

Las sentencias if son prácticamente iguales en C/C++ y en JS. El objetivo y funcionamiento del condicional son iguales, y la sintaxis en ambos casos es la misma:

```
if(condicion) {sentencia}
```

Para el “else” también se mantienen igual:

```
if(condición1) {
     sentencia 1;
     sentencia 2;
} else if(condición 2){
     sentencia 3;
} else {
     sentencia 4;
}
```

En ambos lenguajes las llaves son opcionales si se quiere ejecutar solo una sentencia, y obligatorias si hay una cadena de más de una instrucción.

La razón por la cual son tan parecidas, es que JS comparte raíces con C/C++, por lo que algunas sentencias se mantuvieron. Por esto se dice que la sentencia if en JS es compatible con la de C/C++.