

Taller de Programación

01 - Introducción

Mgs. Lic. Marcos Prunello

2017-08-20

Reseña histórica de la computación

- La palabra **computación** proviene del latín *computare*, cuyo significado es “enumerar cantidades”.
- Designa la acción y efecto de computar, realizar una cuenta, y como tal ha estado presente desde tiempos antiguos.
- La computación era manual, hasta que en 1623 el alemán Wilhelm Schickard inventó la primera calculadora mecánica.
- Los primeros modelos de calculadoras eran mecánicos.



Figure 1: Réplica de la máquina calculadora de Schickard.

Reseña histórica de la computación

- Revolución Industrial → crecimiento de la tecnología → nuevas formas para realizar cálculos
- El matemático británico **Charles Babbage** diseñó dos tipos de máquinas calculadoras:
 - La *máquina diferencial* para crear tablas de funciones matemáticas
 - La *máquina analítica* de uso general, capaz de realizar distintas funciones de acuerdo a cómo fuese “programada”. Era controlada por un patrón de perforaciones hechas sobre una tarjeta que la misma podía leer.
- No llegó a concretar sus diseños en vida pero sentó bases importantes.

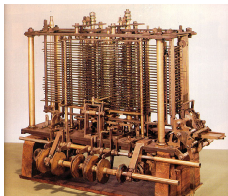


Figure 2: La máquina analítica de Babbage.

Reseña histórica de la computación

- En 1890 **Herman Hollerith** utilizó tarjetas perforadas para automatizar la tabulación de datos para el censo de EEUU y más tarde funda IBM.
- La primera computadora electrónica “programable” fue diseñada por **John Atanasoff** y **Clifford Barry** en 1939.
- Contaba con 300 tubos de vacíos, componentes electrónicos que pueden modificar una señal eléctrica mediante el control del movimiento de los electrones produciendo una respuesta.



Figure 3: Réplica de la máquina calculadora de Schickard.

Reseña histórica de la computación

- La primera computadora electrónica a gran escala fue la **ENIAC** de **Presper Eckert** y **John Mauchly**.
- Tenía más de 18000 tubos de vacío, ocupaba una sala de 9x15 metros y era controlada conectando ciertos cables en un panel
- Otro gran avance se produjo en 1946: **John von Neumann** propuso que los programas y los datos podrían ser representados y guardados en una memoria interna.

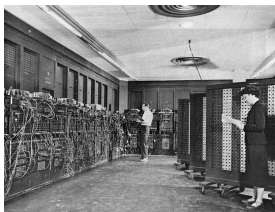


Figure 4: La ENIAC en Filadelfia, Pennsylvania.

Reseña histórica de la computación

- Desde entonces la computación ha evolucionado muy rápidamente
- Se describe al desarrollo de las computadoras modernas con 4 generaciones:
 - **1°G:** computadoras con tubos de vacío para su circuito interno.
 - **2°G:** a partir de 1947, con el desarrollo del *transistor*, mucho más pequeño y consumen menos energía. Igual aún ocupaban mucho espacio.
 - **3°G:** se inició en 1959 con el desarrollo de un circuito integrado (“chip”), una pequeña placa de silicio sobre el cual se imprime un gran número de transistores conectados.
 - **4°G:** comenzó en 1975, se construye la unidad entera de procesamiento de una computadora sobre un único chip de silicio (*microprocesadores*).



Figure 5: De derecha a izquierda: un tubo de vacío, un transistor y un chip.

- Los componentes físicos y materiales que estamos describiendo son sólo una parte de la historia.
- La máquina física que uno puede comprar y llevar al escritorio de casa es un ejemplo de **hardware**. Es tangible.
- Pero para que una computadora pueda cumplir un objetivo debe ser **programada**.
- El acto de programar una computadora consiste en proveer un conjunto de instrucciones - un **programa** - que especifica todos los pasos necesarios para resolver un problema que se le asigna.
- Estos programas generalmente se conocen como **software**
- Es la conjunción de ambos, hardware y software, la que le da vida a la computación.

- A diferencia del hardware, el software es una entidad abstracta, intangible.
- Se trata de una secuencia de pasos simples y operaciones, especificadas en un lenguaje que el hardware puede interpretar.
- En nuestro **Taller de Programación** nos concentraremos en el lado del software, en el diseño de la solución de algún problema y en cómo transmitírsela a la computadora para que la misma pueda ejecutarla.

- Como seres humanos, tenemos incorporada intuitivamente la resolución de problemas cotidianos
- Para intentar afrontar un inconveniente, solemos hacer un proceso rápido de selección e intentamos buscar la opción más favorable.
- Un **algoritmo** es una estrategia consistente de un conjunto ordenado de pasos que nos lleva a la solución de un problema o alcance de un objetivo.
- Características de un algoritmo:
 - Está expresado de manera clara y precisa
 - Es efectivo
 - Es finito

- Los problemas y los algoritmos varían mucho en complejidad.
- Algunos problemas son tan simples que inmediatamente se nos ocurre un algoritmo para su resolución.
- Problemas complejos pueden requerir algoritmos más elaborados
- Se pueden originar distintos algoritmos para solucionar un mismo problema.
- La resolución computacional de un problema consiste de dos etapas básicas:
 - **Diseño algorítmico:** desarrollar un algoritmo, o elección de uno existente, que resuelva el problema.
 - **Codificación:** expresar un algoritmo en un lenguaje que la computadora pueda interpretar.

- Frente a cada problema el primer paso es idear un algoritmo y expresarlo por escrito
- En programación, el lenguaje artificial e informal que usan los desarrolladores en la confección de algoritmos recibe el nombre de **pseudocódigo**.
- No es en sí mismo un lenguaje de programación (la computadora no lo entiende)
- Pero tiene el objetivo de expresar claramente la solución lógica y ser una guía al escribir el programa.

El diseño algorítmico: el pseudocódigo

- El pseudocódigo, como cualquier otro lenguaje, está compuesto por:
 - Un **léxico**
 - Una **sintaxis**
 - Una **semántica**
- Sigue una **estructura secuencial**: define una acción o instrucción que sigue a otra en secuencia.

ALGORITMO: "Ejemplo"

COMENZAR

Acción 1

Acción 2

...

Acción N

FIN

El diseño algorítmico: el pseudocódigo

- Ejemplo: el problema es poner en marcha un auto

ALGORITMO: "Arrancar el auto"

COMENZAR

 INSERTAR la llave de contacto

 UBICAR el cambio en punto muerto

 GIRAR la llave hasta la posición de arranque

 SI el motor arranca

 ENTONCES

 DEJAR la llave en posición "encendido"

 SI NO

 LLAMAR al mecánico

 FINSI

FIN

Lenguajes de programación

- Para que una computadora pueda entender nuestro algoritmo, debemos traducirlo en un **lenguaje de programación**
- Es un idioma artificial diseñado para expresar cálculos que puedan ser llevados a cabo por equipos electrónicos
- Medio de comunicación entre el humano y la máquina.
- Ejemplos: Fortran, BASIC, C++, Java, Python.
- En este curso usaremos SAS/IML.



Figure 6: Distintos lenguajes de programación y sus logos.

Codificación: creación y edición de programas

- Cada una de las acciones que componen al algoritmo son codificadas con una o varias **instrucciones** o **sentencias**, expresadas en el lenguaje de programación elegido
- Su conjunto constituye un **programa**.
- El programa se guarda en un **archivo** cuyo nombre tiene:
 - Una **raíz**: describe el contenido
 - Una **extensión** es indicativa del uso del archivo
- Ejemplo: `miPrimerPrograma.sas`
- El proceso de ingresar o modificar el contenido de un archivo se denomina **edición**.

Tipos de lenguajes de programación

- **Lenguajes de bajo nivel:** más próximos a la arquitectura del hardware, más rígidos y complicados de entender para nosotros.
- **Lenguajes de alto nivel:** más cercanos a los programadores y usuarios, diseñados para que sea fácil para los humanos expresar los algoritmos sin necesidad de entender en detalle cómo hace exactamente el hardware para ejecutarlos.
- El lenguaje que utilizaremos en este taller, SAS/IML, es de alto nivel.
- Para que un programa escrito en un lenguaje de alto nivel pueda ser ejecutado, se necesita de **compiladores** o **intérpretes**
- Traducen al lenguaje de bajo nivel que es apropiado para el hardware que se dispone.

- **Errores sintácticos:**

- Los lenguajes de programación tienen su propio vocabulario y su propia sintaxis.
- Cuando corremos un programa, el compilador primero chequea si es sintácticamente correcto y muestra un mensaje de error si no lo es.

- **Errores lógicos:**

- Suelen ser más problemáticos.
- Ejemplo: el programa compila sin errores pero arroja resultados incorrectos o ningún resultado.
- Hay que revisar el programa para encontrar algún error en la lógica del mismo.
- Estos errores se llaman **bugs** y la corrección de los mismos **debugging** (depuración).

A la hora de resolver un problema computacional, seguiremos los siguientes pasos:

- Analizar el problema que vamos a resolver.
- Imaginar una solución (algoritmo).
- Traducir la solución a pseudocódigo.
- Implementar en un lenguaje de programación todo lo analizado.
- Compilar o correr el programa.
- Realizar pruebas de ejecución.
- Corregir los errores que haya.

El programa “Hola mundo”

- C es uno de los más exitosos en la historia de la programación, creado en 1972 por Dennis Ritchie.
- En el libro de presentación, *The C Programming Language* (1978), los autores proponen lo siguiente en la primera página del primer capítulo:

“The only way to learn a new programming language is by writing programs in it. The first program to write is the same for all languages:

```
PRINT THE WORDS  
    hello, world
```

This is the big hurdle; to leap over it you have to be able to create the program text somewhere, compile it successfully, load it, run it, and find out where your output went. With these mechanical details mastered, everything else is comparatively easy.”

El programa “Hola mundo”

- El libro continúa con las cuatro líneas del programa “hello world”
- El consejo está bueno: el primer programa que uno escriba debe ser lo más sencillo posible para darnos la oportunidad de entender el mecanismo del proceso de programación.
- Esto se convirtió en una tradición: al aprender un nuevo lenguaje el primer objetivo es escribir un programa para saludar al mundo.
- Vamos a probarlo en SAS/IML, el lenguaje que utilizaremos nosotros!

El programa “Hola mundo” en SAS/IML

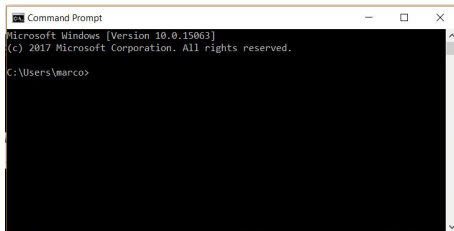
- 1 Abrir el Block de Notas (o cualquier otro programa para escribir un archivo de texto)
- 2 Escribir las sentencias:

```
proc iml;  
    print "hola mundo";  
quit;
```

- 3 Guardarlo con un nombre que describa su contenido y que su extensión indique que es un programa de SAS, por ejemplo, `saludarAlMundo.sas`.

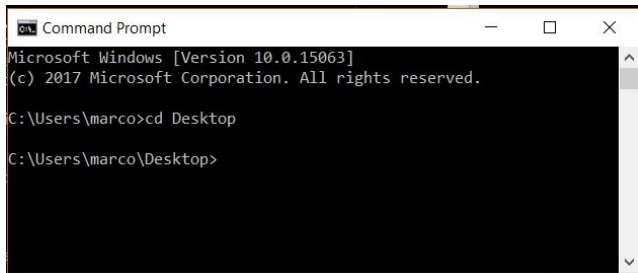
Interfaz de línea de comandos

- Ya tenemos nuestro programa escrito en SAS/IML y guardado en un archivo.
- Ahora tenemos que pedirle a la computadora que lo ejecute.
- Para eso vamos a abrir la **interfaz de línea de comandos** de Windows, que es un método que permite a los usuarios dar instrucciones a la compu por medio de una línea de texto simple.
- En Windows se la conoce como *símbolo del sistema* o *cmd* (por “command”).
- La abrimos escribiendo “cmd” en el campo de búsqueda de la barra de tareas.



Correr nuestro programa

- Ahora tenemos que indicarle a la compu dónde está el archivo con el programa que queremos correr.
- Lo hacemos indicándole el camino completo hasta la carpeta donde lo guardamos.
- En mi caso, lo tengo en el Escritorio, y me dirijo hasta allí con el comando “cd” (*change directory*):



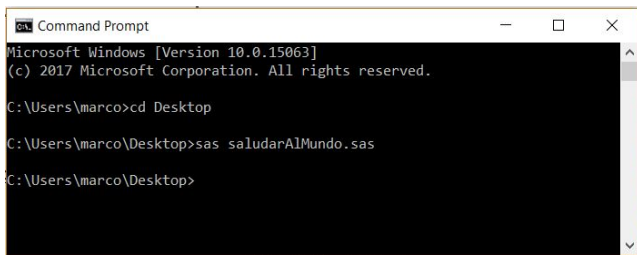
```
Command Prompt
Microsoft Windows [Version 10.0.15063]
(c) 2017 Microsoft Corporation. All rights reserved.

C:\Users\marco>cd Desktop

C:\Users\marco\Desktop>
```

Correr nuestro programa

- Ahora que el sistema está ubicado en el mismo lugar que mi programa, le digo que lo corra.
- Se lo indico con el comando “sas” (indicando que el lenguaje usado es SAS), seguido por el nombre del archivo a ejecutar, `saludarAlMundo.sas`:



```
Command Prompt
Microsoft Windows [Version 10.0.15063]
(c) 2017 Microsoft Corporation. All rights reserved.

C:\Users\marco>cd Desktop

C:\Users\marco\Desktop>sas saludarAlMundo.sas

C:\Users\marco\Desktop>
```


- Como resultado, en la misma carpeta donde guardamos nuestro script, ahora tenemos dos archivos:
 - **saludarAlMundo.lst**: contiene el resultado o “output” de nuestro programa (el saludo “hola mundo”).
 - **saludarAlMundo.log**: contiene la descripción del procedimiento ejecutado, incluye, si hubo, explicaciones sobre los errores encontrados.
- Estos archivos son archivos de texto comunes, que podríamos abrir, por ejemplo, con el Block de Notas (botón derecho, abrir con. . .)

- ¡Bien! Corrimos nuestro programa, abarcando todo esto:
 - Escribimos una sentencia en lenguaje SAS/IML.
 - Guardamos el archivo con extensión .sas para indicar su lenguaje.
 - Abrimos el símbolo del sistema que es el que puede ejecutar nuestro programa.
 - Le indicamos al sistema dónde estaba nuestro programa y que lo ejecute con SAS.
 - Obtuvimos un resultado. . .
 - Y quedamos bien con todo el mundo!!
- Ahora quedemos bien con alguien en particular!
- Repitamos el ejercicio pero esta vez para saludar a una persona por su nombre.

Con los pies sobre la tierra...

- Con el ejemplo anterior pudimos tener un vistazo de los pasos a seguir para poder correr un programa.
- Pero la realidad es que nos queda mucho por aprender antes de estar en condiciones de crear programas más complejos.
- Y para facilitar ese aprendizaje vamos a usar SAS desde su interfaz gráfica que ya todos conocemos.
- Además, antes de intentar escribir un programa en SAS/IML, vamos a pensar con detenimiento el algoritmo que resuelve al programa en cuestión y expresarlo en pseudocódigo.

- **SAS** es un paquete de programas orientados al análisis estadístico desarrollado por **SAS Institute** a finales de los años sesenta.
- Este instituto se inició a finales de los años sesenta como un proyecto en la Universidad de Carolina del Norte para crear un *sistema de análisis estadístico* (**S**tatistical **A**nalysis **S**ystem) originalmente utilizado por los departamentos de Agricultura de algunas universidades.
- En 1976 se convirtió en una compañía y privada y desde entonces tomó relevancia y gran prestigio en la comunidad estadística internacional.

- IML son las siglas de *Interactive Matrix Language*.
- Es decir, IML es un lenguaje de programación que hace foco en la utilización y manejo de vectores y matrices, permitiendo hacer con ellos cálculos de alto nivel.
- Permite interactuar con otros procedimientos de SAS, acceder a archivos, y crear gráficos entre otras cosas.
- Incluye un variado conjunto de funciones y operadores para asistir en la programación.

- SAS no es un software libre y como tal para poder instalarlo y hacer uso del mismo se debe adquirir una licencia.
- Sin embargo, SAS ofrece una versión gratuita pensada para estudiantes universitarios llamada SAS University Edition.
- El link anterior lleva a la página oficial que detalla todos los pasos para poder instalar y usar esta versión gratuita.
- Como alternativa, este video también provee instrucciones detalladas para Windows.