Taller de Programación

01 - Introducción

Mgs. Lic. Marcos Prunello

2018-07-29

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.

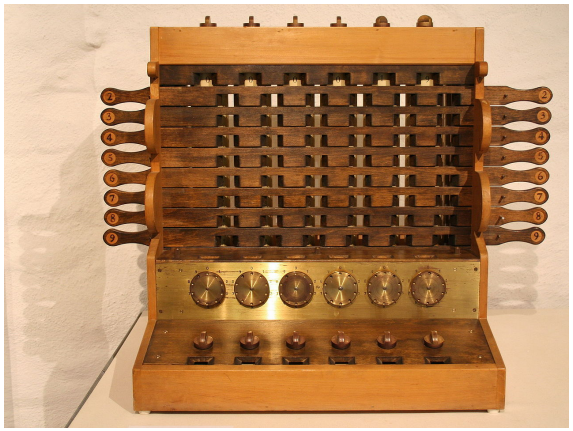

Figure 1: Réplica de la máquina calculadora de Schickard.

Reseña histórica de la computación

- Revolución Industrial $\rightarrow$ crecimiento de la tecnología $\rightarrow$ 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.

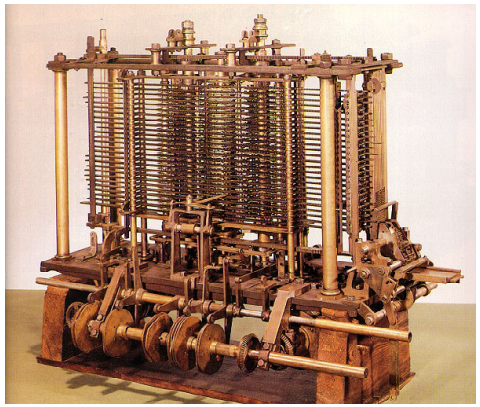

Figure 2: La máquina analítica de Babbage.

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.

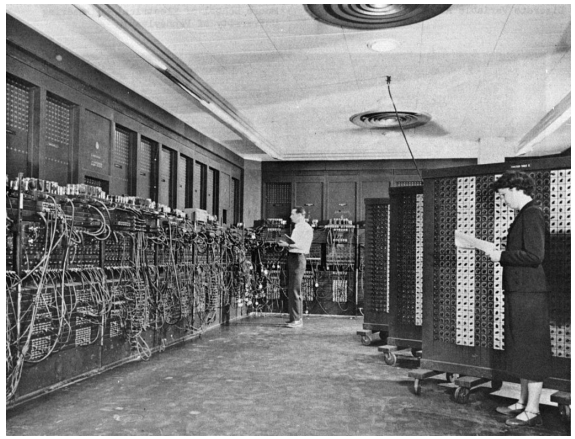

Figure 4: La ENIAC en Filadelfia, Pennsylvania.

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*).

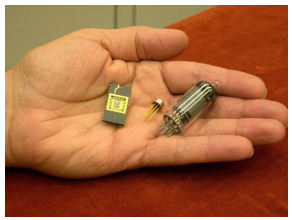

Figure 5: De derecha a izquierda: un tubo de vacío, un transistor y un chip.

Software y hardware

- 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.

Software y hardware

- 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.

Problemas y algoritmos

- 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

Problemas y algoritmos

- 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.

El diseño algorítmico

- Frente a cada problema el primer paso es idear una 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 cómputos 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 **raiz**: 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 de programación y depuración

- **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).

Resumiendo

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.