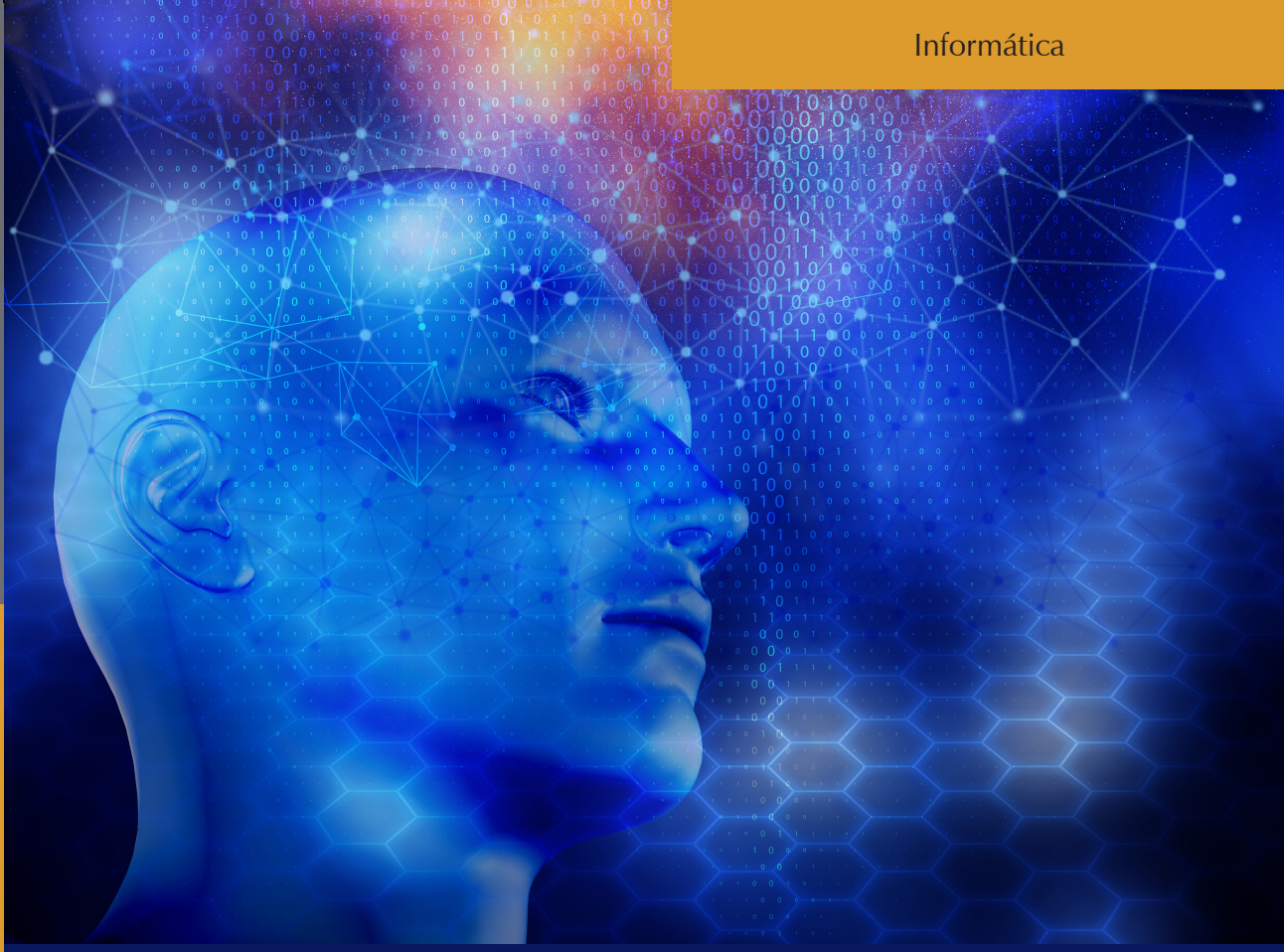
Informática



Lógica de programación

Omar Iván **Trejos** Buriticá

ediciones de la U

Informática

Lógica de Programación

Omar Iván **Trejos** Buriticá

ediciones de la U

Trejos Buriticá, Omar Iván
Lógica de programación -- 1a. edición. Bogotá : Ediciones de la U, 2017.
432 p. ; 24 cm.
ISBN 978-958-762-720-6 e-ISBN 978-958-762-721-3
1. Programación 2. Lógica 3. Variables, constantes y operadores
4. Algoritmos 5. Ciclos 6. Matrices I. Tít.
519.7cd 21 ed.

Área: Informática
Primera edición: Bogotá, Colombia, noviembre de 2017
ISBN. 978-958-762-720-6

Coordinación editorial: Adriana Gutiérrez M.
Carátula: Ediciones de la U
Impresión: Digiprint Editores SAS
Calle 63 #70D-34, Pbx. (57+1) 7217756

Impreso y hecho en Colombia
Printed and made in Colombia

Apreciad@ cliente:

Es gratificante poner en sus manos estas obras, por esta razón le invitamos a que se registre en nuestra web: **www.edicionesdelau.com** y obtenga beneficios adicionales como:

- **Complementos digitales de las obras**
- **Actualizaciones de las publicaciones**
- **Interactuar con los autores a través del blog**
- **Información de nuevas publicaciones de su interés**
- **Noticias y eventos**

Adquiere nuestras publicaciones en formato e-book 

Visítanos en:

www.edicionesdelau.com

Sus pedidos a:
Carrera 27 # 27-43 • Barrio Teusaquillo
PBX. (57-1) 3203510 • (57-1) 3203499 • Móvil: 310 - 6256033
comercial@edicionesdelau.com - gerencia@edicionesdelau.com
Bogotá - Colombia

Av. Coyoacán 1812 A. Acacias Benito Juárez C.P. 03240
PBX. (52) 55-63051703 • Cel. 044 5544439418
janethcr@gruporamadelau.com
México D. F. - México

Contenido

A Natalia y a Juan José,
¡¡¡mi maravilloso todo!!!

Introducción

Durante muchos años he dedicado gran parte de mi tiempo no solo a la enseñanza de la Lógica de Programación, sino al análisis de la enseñanza de dicha Lógica debido, precisamente, a que me he encontrado con que muchas personas confunden la Programación con la Lógica de Programación. La primera involucra el conocimiento de técnicas e instrucciones de un determinado lenguaje a través de los cuales se hace sencillo lograr que el computador obtenga unos resultados mucho más rápido que nosotros. La segunda involucra, de una manera técnica y organizada, los conceptos que nos permiten diseñar, en términos generales, la solución a problemas que pueden llegar a ser implementados a través de un computador.

El estudio de la Lógica de Programación no exige ningún conocimiento previo de computadores ni de tecnología en general; tampoco exige la presencia de algún lenguaje de programación específico, aunque no puedo negarle que este podría permitirle, solo después de que usted maneje bien los conceptos de Lógica de Programación, la implementación de las soluciones lógicas a sus objetivos.

Fueron muchos los alumnos que con el tiempo me fueron solicitando que les enseñara cuáles eran los conceptos realmente básicos para aprender a programar, o sea, aquellos conceptos con los cuales es suficiente para enfrentarse a cualquier lenguaje de programación o, mejor aún, enfrentarse a lograr cualquier objetivo a través de un computador. Poco a poco, fui buscando soluciones a las preguntas que mis alumnos me planteaban y veía que, en sus dudas, siempre estaba presente la búsqueda de conceptos esenciales que los liberara de las ataduras que tienen los lenguajes de programación cuando estos son lo primero que se conoce en computadores.

Luego de muchos años de estudio de estos factores, pude condensar en este libro los que, considero, son los conceptos fundamentales para aprender realmente a programar, o sea, lo que he llamado *la esencia de la Lógica de Programación*, pues busco que usted conozca estos elementos conceptuales y, luego

de dominarlos, se enfrente sin ningún problema no solo a cualquier objetivo que pueda ser alcanzable a través de computadores, sino además a cualquier lenguaje de programación.

Puedo garantizarle que, si usted lee este libro hoja por hoja y desarrolla los ejercicios aquí planteados, al llegar al final del mismo, podrá entender que programar no es más que buscar soluciones muy lógicas utilizando unos conceptos muy sencillos. Espero, pues, que este libro cumpla el objetivo planteado, pues pensando en usted fue como se concibió. No se vaya a afanar por leerlo de una sola vez; tómese su tiempo para razonar y asimilar los conceptos que aquí se plantean. No olvide que, para resolver problemas computables, usted no aplica su propia lógica, sino que toma prestada una lógica que no es la natural.

Este libro en ninguna de sus partes le mostrará conceptos complejos, debido precisamente a que la lógica de programación es la unión de muchos (pero muchos) conceptos sencillos para el diseño de soluciones muy (pero muy) lógicas.

Omar Iván Trejos Buriticá, PhD

¿Cómo usar este libro?

En primera instancia, le recomiendo que lo lea pausadamente. No se afane en avanzar; la apropiación y asimilación de los conceptos pertinentes a la lógica de programación implica tiempo, pues no son exactamente los mismos conceptos que subyacen a la lógica natural y deliberativa que tenemos los seres humanos.

Lea y vuelva a leer, piense en lo que ha leído y, ante todo, realice (o, por lo menos, intente realizar) los ejercicios propuestos. Pregunte cuando tenga dudas y siempre propóngase terminar los ejercicios. No los deje a medio camino, pues cada vez que usted lleve un ejercicio hasta el final verá cómo su lógica humana se amplía para acudir a la lógica computacional en los casos donde corresponda.

Para el profesor

Utilice este libro al ritmo que sus estudiantes le permitan. Usted y yo sabemos que lo más importante no es avanzar en un contenido, sino que los estudiantes realmente aprendan lo que se puede asimilar de ese contenido. Recuerde que asimilar una nueva lógica toma tiempo y lo que para algunos puede ser muy obvio para otros puede ser complejo. Esta consideración será de gran utilidad para intentar comprender los diferentes niveles de aprendizaje y apropiación de la lógica por parte de cada uno de los estudiantes que son y serán nuestra razón de ser.

Para el estudiante

Siga el ritmo que le indique el profesor. Revise los ejercicios resueltos, resuelva los ejercicios propuestos y, siempre, sin excepción, pregunte. No se quede con dudas. Por simples que le parezcan, recuerde que, cuando se trata de Lógica, las dudas son también lógicas, pero si no se resuelven, simplemente van quedando lagunas cuya resolución posteriormente puede llegar a ser más

compleja. Comparta soluciones con sus compañeros. Es muy enriquecedor alimentarse de la lógica de los demás y aportarles lo que, desde nuestra lógica, se pueda aportar. La forma excelsa para aprender a nadar es nadando; asimismo, el camino óptimo para asimilar la lógica de programación es practicando, practicando y practicando.

¿Qué es programar?

Programar es encontrar soluciones, basadas en lógica de programación, que permiten que el computador alcance por nosotros un determinado objetivo. El artífice de que el computador logre dichos objetivos es la persona que lo programó.

Capítulo 1

La Lógica

1.1. Hablando de Lógica

Recuerdo que, en mi niñez, alguna vez me abroché mal la camisa, en un momento en que toda mi familia estaba afanada por salir. Una familiar me vio la camisa mal abrochada y me dijo fuertemente que me había abrochado mal la camisa, que si era que yo no tenía lógica. Luego de acomodármela adecuadamente, o sea, de manera que cada botón coincidiera con su respectivo ojal, comencé a pensar que era posible que no tuviera lógica, pues me parecía sorprendente que no me hubiera dado cuenta de que para que la camisa estuviera colocada correctamente solo hay una forma y es que coincidan par botón-ojal. Llegué a otra conclusión y es el hecho de que es más fácil ponerse bien una camisa que ponérsela mal o, dicho en otras palabras, es muchísimo más fácil colocársela correctamente en lo que a botones y ojales corresponde.

Fui creciendo y poco a poco me di cuenta de que son muchas las cosas que, pareciendo obvias, por un extraño error no hacemos bien y vuelve a mi memoria la voz de mi familiar diciendo "¿¿¿Usted no tiene lógica o qué???". Estudié mi carrera universitaria Ingeniería de Sistemas porque allí encontré por qué era tan importante aquello de la lógica. Luego de buscar muchos significados de la palabra lógica, llegué a una que al fin me convenció. Le pregunté a una amiga "¿Qué es la lógica...?". Ella respondió en un lenguaje muy común: "*Pues lógica es... es.... es.... es como algo muy lógico*". Su respuesta no me satisfizo, pues había incluido en la definición el término que quería definir, lo cual significa que no me había dicho nada. Cuando pregunté por qué era difícil definirlo, ella respondió: "*No es fácil definir algo tan lógico*". Ella tenía clara la idea del significado, simplemente no era capaz de definirlo.

Luego le pregunté a don Luis, un viejo tendero que por diez años lo había visto llegar todas las mañanas a abrir su tienda desde donde atendía a su clientela.

Él me respondió: "*Lo único que puedo decir es que lo que es lógico es todo aquello que nunca es ilógico*". Esa definición me pareció bastante racional, pero seguía siendo lejana de lo que yo había esperado. Yo veía que el proceso de abrir su tienda tenía unos pasos definidos y siempre los hacía de forma bastante lógica.

Después le pregunté a un profesor de la materia Español y él me entregó una excelente definición de esas tomadas de un diccionario: "*Lógica es la rama del conocimiento que nos permite definir que algo está respaldado por la razón como bien deducido o bien pensado*". Para mí era una buena definición y me bastaba con que apareciera en el diccionario *Pequeño Larousse* para no discutirla. Sin embargo me exigía más reflexiones de las necesarias para poder entenderla, pues me parecía increíble que la definición de la palabra lógica fuera tan compleja, o sea, que su definición no fuera lógica. Eso mismo me había motivado a buscar una definición sencilla y lógica que no me exigiera muchas reflexiones adicionales.

Buscando una definición que me dejara satisfecho, fui a preguntarle a un matemático. Yo suponía que él podría definir qué era la lógica. Cuando le pregunté, me respondió: "*Lógica es la ciencia que estudia la estructura, fundamentos y uso de las expresiones del conocimiento humano*". Esa era una definición muy exacta pero, al igual que la definición del *Pequeño Larousse*, me exigía demasiados razonamientos como para poder digerirla.

Me animé a preguntarle a alguien, un desconocido, qué era la lógica y su respuesta desprevenida me gustó porque la entendí fácilmente: "*La lógica es como una serie coherente de ideas y razonamientos*". Compartí su definición y me pareció apropiada. Además que pude descubrir que todas las personas a quienes les preguntaba tenían muy claro el concepto de lógica, así algunas de esas personas no la pudieron definir de manera clara.

Finalmente, busqué a un campesino al que los avances tecnológicos no lo hubieran tocado aún. Alguien para quien el mundo moderno era un conjunto de problemas en vez de soluciones. Le pregunté: "¿Qué es la lógica?" y mirándome con extrañeza me dijo: "*Mire patrón, pues eso es la forma más OBVIA y FÁCIL de hacer cualquier cosa*". Entonces vi que, de todas las definiciones que había recogido, esta era la que me parecía más lógica. Concluí que eso es la LÓGICA.

Los libros de tecnología citan que, para resolver un problema a través del computador, se requiere tener muy buena lógica. Creo que la base para ello es ser muy lógicos, o sea, poder vislumbrar el camino más obvio y más fácil para lograr un objetivo. Este libro busca orientar su lógica natural de manera que para usted sea muy sencillo hablar de la lógica computacional.

1.2. Fundamentos conceptuales

Vamos a comenzar por plantear una opinión sobre María... ¿y quién es María? Es la figura que nos va a acompañar en esta explicación. Se lo voy a decir muy claro: "María es alta". Inmediatamente usted, querido lector, se imaginará una mujer con más de 1,70 m de estatura, que puede ser más alta que usted o que, por lo menos, es de su misma estatura (si es que usted se considera alto). Apenas yo digo "María es alta", usted deberá hacer unos razonamientos lógicos y concluyentes para captar lo que yo le quise decir. ¿Qué fue lo que yo le describí de María? Muy fácil, describí un **atributo** de ella. ¿Qué es un atributo? Es una característica que identifica a un **ente informático**. ¿Y qué es un ente informático? Todo aquello que se puede describir basándose en sus características.

¿Qué características tiene un atributo? La primera consiste en que obedece a una serie de razonamientos humanos, lo cual quiere significar que debe existir todo un juego de razonamientos previos. La segunda característica es que es muy relativo. Si María mide 1,65 y vive en Occidente, puede no ser tan alta; si ella vive en Europa, sería una persona baja y si de pronto ella vive en Oriente, sería una persona notoriamente alta. Los atributos siempre van a estar sujetos a la mirada relativa con la que observa quien emane el concepto. Similar es el caso que se presenta cuando un hombre dice que una mujer es muy hermosa, pues lo que para él es una mujer hermosa, es posible que para otros no lo sea tanto.

Como se puede ver, estas dos características, en unión con un conjunto de conceptos y vivencias provenientes de la cultura de la región en donde hemos crecido, logran que se afiance más lo relativo de un atributo. Debido a esta relatividad conceptual sobre los atributos, estos son inmanejables porque van a depender del observador que los esté usando.

Se ha hecho necesario a través de la Historia que los atributos sean medidos a través de escalas, ya que esto los hace manejables y dejan de ser relativos (sin decir con esto que se vuelvan absolutos). Es por ello que surge la gran vedete de la Informática que es el **dato**. Nuestra frase inicial "María es alta" se puede cambiar a decir "María mide 1,73 m". A pesar de que los razonamientos y las conclusiones podrían ser los mismos, se pueden dejar al libre concepto de quien los observe. ¿Qué es un dato? Es un atributo "codificado" en unos términos que sean entendibles a un sistema de información, que sean manejables y comparables y que son, en gran medida, absolutos.

Un atributo es "codificado" si ha sido convertido a una escala determinada para poder ser más manejable, lo cual quiere decir que se puede operar con otros

atributos de la misma escala, es decir, se pueden realizar comparaciones y obtener resultados de ellas. Un dato en solitario no significa nada, excepto que se tenga claro cuál es el atributo que se está describiendo. Si yo le dijera: "Amigo lector, le cuento que el dato es 4", ¿qué pensaría usted que significa este dato? La cantidad de carros del autor o la cantidad de libros del autor o la cantidad de amigos del autor o... realmente usted no tendría certeza de su significado. Viene un concepto que aclara las cosas.

Nuestra frase inicial "María es alta", que luego se convirtió en "María mide 1,73 m", podríamos ahora plantearla como "La estatura de María es 1,73". En este momento, ya tenemos identificado de manera clara y con un nombre el atributo que se intenta describir. Este es el concepto de **campo**, que es el nombre que se le coloca a un dato para identificar el atributo que se propone describir. Así, nuestra frase "La estatura de María es 1,73" presenta tres campos bien identificados. El 1º de ellos es la estatura, campo con el que se ha estado realizando toda la explicación; el 2º corresponde al nombre de la persona de quien se está hablando, que es María en este ejemplo, y el 3º es el sexo, pues se puede suponer que María es de sexo femenino. La tabla 1 presenta la información de manera organizada.

Tabla 1. Información organizada

Nombre de la persona	Estatura de la persona	Sexo de la persona
María	1,73 m	Femenino

Se tiene aquí un conjunto de campos de forma que en cada campo está consignado un dato, en donde todos los datos pertenecen a un mismo ente informático. Con esto le acabo de entregar la definición de lo que es un **registro.** En esas condiciones se le puede colocar un nombre identificador al registro del ejemplo y lo vamos a llamar *Persona*. También se le pueden adicionar otros campos y llenarlos con datos del mismo ente informático, tal como se muestra en la tabla 2.

Tabla 2. Información organizada con más campos

Registro *Persona*					
Nombre	**Estatura**	**Sexo**	**Fecha de nacimiento**	**No. cédula**	**Salario**
María	1,73 m	Femenino	21-ago-78	42.522.301	560.000,00

Se puede pensar en organizar de una forma más apropiada la información que corresponde a María buscando que sea más presentable y más manejable. Una versión del registro de estudio mejor organizado se presenta en la tabla 3.

Tabla 3. Registro organizado

Registro *Persona*					
No. cédula	**Nombre**	**Sexo**	**Fecha de nacimiento**	**Estatura**	**Salario**
42.522.301	María	Femenino	21-ago-78	1,73 m	560.000,00

¿Cuántos campos pueden pertenecer a un registro? Todos los que usted necesite, es decir, todos los campos en donde los datos sean útiles para usted. Una característica adicional que debe cumplir un registro es que debe ser manejado como una sola unidad, es decir, que todos los campos que lo conforman se encuentren en el mismo lugar físico o lógico de manera que pueda ser manipulado como un todo. Ahora, ¿qué sucedería si además de los datos de María tenemos también los datos de Luis, Pedro, Aníbal, Marta, Elena y Julián obteniendo de cada uno los mismos campos que obtuvimos de María? Pues simplemente que se ha conformado un **archivo**, que es un conjunto de registros que tienen la misma estructura y que se puede manejar como una sola unidad. Hablar de registros con la misma estructura quiere decir que tienen los mismos campos pero no los mismos datos. La tabla 4 presenta la información de varias personas.

Tabla 4. Archivo de varias personas

Registro *Persona*					
No. cédula	**Nombre**	**Sexo**	**Fecha de nacimiento**	**Estatura**	**Salario**
42.522.301	María	Femenino	21-ago-78	1,73 m	960.000,00
10.544.676	Luis	Masculino	20-ene-75	1,60 m	800.000,00
16.432.435	Pedro	Masculino	25-feb-70	1,55 m	980.000,00
10.398.789	Aníbal	Masculino	18-abr-55	1,98 m	890.000,00
41.884.556	Marta	Femenino	20-jul-79	1,77 m	900.000,00
38.756.986	Elena	Femenino	16-sep-65	1,58 m	999.000,00

Si lo que se necesita es almacenar tanta información que debemos guardarla en varios archivos pero que estén interrelacionados, entonces se está hablando de una **base de datos**, que es un conjunto de archivos o tablas organizados bajo unas técnicas específicas de normalización.

Estas definiciones nos han llevado desde un concepto completamente humano, como es el atributo, hasta un concepto absolutamente técnico, como es la base de datos. Si miramos el trasfondo de toda esta secuencia, podemos descubrir cuál es su objetivo. El objetivo es hablar de **información.** ¿Cómo se puede definir la información? Información es un conjunto de datos suficientemente organizados y entendibles (algunas veces se organizan a través de la tecnología).

¿Qué es la **Informática**? Es la ciencia que estudia, aplica y optimiza el tratamiento eficiente de la información. ¿Qué significa aquello de tratamiento eficiente de la información? Simplemente significa que es una ciencia que se ocupa de que la información cumpla con dos objetivos:

a. La veracidad: "Toda información debe ser cierta (es decir, veraz)". A usted de nada le sirve que vaya al banco y solicite su saldo y sin ninguna demora le den un saldo que no es el suyo.

b. La oportunidad: "Toda información debe llegar en el momento indicado (o sea, oportunamente)". A usted no le sirve que en el banco le digan que su verdadero saldo se lo entregan en 8 meses.

¿Por qué cada vez que se habla de Informática se relaciona con computadores? Pues porque en la actualidad los computadores son los dispositivos, aparentemente, mejores para cumplir con el objetivo de la oportunidad, ya que trabajan a unas altísimas velocidades. ¿Y quién se encarga de la veracidad? De ella se encarga el ser humano, que es quien planea, organiza y programa lo que el computador va a entregar como información.

1.3. Evaluación

- ¿Qué es la lógica?
- ¿Qué es un ente informático?
- ¿Qué es un atributo?
- ¿Qué es un dato?
- ¿Qué es un campo?
- ¿Qué es un registro?
- ¿Qué es un archivo o tabla?
- ¿Qué es una base de datos?
- Defina la palabra Informática.
- ¿Qué es la veracidad y la oportunidad?

1.4. Taller

- Escriba en una hoja de papel un dato suyo.
- Póngale un nombre a ese dato.
- Escriba su nombre (sin apellidos).
- Escriba, al lado de su nombre, sus apellidos.

- Escriba su estatura.
- Escriba su edad y su fecha de nacimiento.
- Escriba su número de identificación.
- Organice toda esa información en forma de registro.
- Escriba los mismos datos de cuatro amigos suyos.
- Organice la información en forma de tabla.

Capítulo 2

Metodología para solucionar un problema computable

Siempre que vamos a resolver un problema, nos enfrentamos con la dificultad de tener que encontrar precisamente eso: una solución. Pocas veces nos detenemos a pensar que existe un camino estructural que nos permite resolver cualquier problema (en términos generales) teniendo, como es obvio, que entrar en la minucia del detalle dependiendo del problema.

2.1. El objetivo

¿Cuál es el primer paso que debemos dar cuando nos enfrentamos a un problema? Lo primero que debemos tener muy claro es: ¿cuál es el problema a resolver? Es evidente que no podemos avanzar hacia la casa de un amigo si no sabemos en donde vive porque las posibilidades de que lleguemos son casi nulas. De manera que lo primero a conocer muy bien es el problema cuya solución la vamos a denotar con el nombre **OBJETIVO**.

Tener claro el objetivo nos va a permitir obtener dos beneficios que, a la postre, serán más grandes de lo que podemos pensar:

a. Nos permite saber hacia dónde vamos.
b. Nos permite saber dónde debemos parar.

Estas dos definiciones parecieran ser lo mismo pero no lo son. Usted puede tener muy claro hacia dónde va pero podría no saber hasta dónde debe llegar o, dicho en otras palabras, podría no saber en dónde debe parar o podría saber en dónde debe parar pero no tener ni idea por cuál ruta llegar. El OBJETIVO se ha de convertir en la razón de ser en la solución de un problema.

Alguna vez, antes de irme a estudiar a la ciudad de Bogotá, mi padre, en una de esas tardes en las cuales se sentaba a aconsejarme, me dijo: "*Te vas a ir a Bogotá con el objetivo de estudiar. Vas a tener toda la libertad del mundo para hacer todo lo que quieras pero, eso sí, independiente de todo lo que hagas en la capital, métete en tu cabeza que la clave del éxito para cualquier cosa en la vida es* ***no perder de vista el objetivo*** *cualquiera que este sea*". Desde allí entendí que realmente tener un objetivo claro, verdaderamente claro, exageradamente claro, es más importante que cualquier otra cosa, porque gracias a ello puede uno ir detrás de dicho objetivo hasta lograrlo. Cada paso que se dé, debe ir en pos del OBJETIVO.

En nuestro caso, podemos decir que, para llegar a la solución de un problema, la clave está en **tener muy claro cuál es el objetivo** y **no perderlo nunca de vista**. Tal vez usted tendrá alguna inquietud en cuanto a la insistencia de este tópico, pero la realidad es que muchas veces creemos tener claro el objetivo y, solo cuando nos empeñamos en lograrlo, vemos que no era así.

2.2. El algoritmo

Tener claro el objetivo nos permite algo adicional. Aquí voy a utilizar una frase que, aunque un poco romántica, nos va a ilustrar claramente: el OBJETIVO es el faro que, solo cuando está bien claro, nos ilumina el camino para lograrlo. Cuando el objetivo está suficientemente claro, podemos vislumbrar un camino lógico para llegar hasta él. Ese camino lógico va a tener un nombre, dada la orientación de este libro, y ese nombre es ALGORITMO.

¿Qué es un ALGORITMO? Es un conjunto de pasos secuenciales y ordenados que permiten lograr un objetivo. Que sean pasos secuenciales significa que deben ser ejecutados uno después de otro y que sean pasos ordenados quiere decir que deben llevar un orden que, en algunos casos, podría ser obligatorio. Como puede notar, el ALGORITMO permite lograr un OBJETIVO, o sea, que este es el camino que necesitamos para lograrlo.

De nuevo, ¿cuándo podemos vislumbrar claramente el algoritmo? Solo cuando el OBJETIVO está realmente claro. Siempre que usted, en el desarrollo de la solución de un problema, vea que en algún momento no sabe para dónde coger, no sabe qué hacer o se siente perdido, no busque más, simplemente quiere decir que realmente usted no tenía tan claro el objetivo como había pensado.

¿Cómo se estructura un objetivo? Muy sencillo; esto le va a parecer muy obvio, pero aun así se lo voy a decir. Un algoritmo se estructura comenzando en un

inicio y terminando en un *fin*. Mucho de lo que encuentre en este libro notará que es exageradamente lógico, pero no olvide que ese es el tema que evoca el título de este libro.

Veamos, entonces, un ejemplo: desarrollar un algoritmo que nos permita adquirir el libro ***El coronel no tiene quien le escriba*** de Gabriel García Márquez.

Objetivo: adquirir el libro *El coronel no tiene quien le escriba* de Gabriel García Márquez. Mucha atención al objetivo. Solamente es adquirirlo, en ningún momento el objetivo es leerlo o resumirlo ni nada, solamente adquirirlo.

Algoritmo: salimos del lugar en donde estemos y nos dirigimos hacia una librería. En caso de que ya estemos en una librería, solicitamos si tienen el libro. Si lo tienen, lo adquirimos y si no lo tienen, vamos a otra librería en donde repetimos el proceso.

Explicado así, el algoritmo no va a pasar de ser un breve texto explicativo que nos va a permitir lograr algo y que, en este caso, es la adquisición de un libro determinado. Pero podríamos organizar este algoritmo de manera que fuera un poco más estético y, por qué no decirlo, un poco más entendible, comenzando por el hecho de que esta vez le vamos a colocar un nombre al algoritmo y que lo vamos a generalizar para conseguir cualquier libro siempre y cuando esté completamente definido.

Algoritmo Adquisicion_Libro
Inicio
1. Saber cuál es el libro que se quiere adquirir
2. Desplazarnos hacia una librería
3. Preguntar si tienen el libro que necesitamos
4. Si lo tienen
adquirirlo y parar allí (dentro de este algoritmo)
Si no lo tienen
ir al paso 2
Fin

Note algunas puntualizaciones al respecto de este algoritmo:

a. Casi todas las líneas van numeradas, pero no todas.

b. En la línea 1, se debe cumplir esa orden para poder continuar con el resto del algoritmo, porque se asume en el algoritmo que no solo se pasa por encima de las líneas, sino que se realizan las tareas allí indicadas.

c. Si realizamos todos los pasos que indica este algoritmo, podremos obtener el libro que sea porque la connotación de este es absolutamente genérico sin restricciones, ya que en ningún momento se está diciendo que nos desplacemos hacia una librería que quede en la ciudad.

d. Si luego de recorrer todas las librerías de todos los países de todo el mundo vimos que no pudimos conseguir el libro, entonces podemos obtener dos conclusiones: una es que el libro que buscábamos no lo tiene ninguna librería porque está agotado y la otra es que el libro es posible que nunca haya existido.

e. Si probamos este ejemplo con el libro en mención (o sea *El coronel no tiene quien le escriba*) tendremos un alto porcentaje de seguridad de que lo conseguiremos a menos que esté agotado.

Este tipo de algoritmos son conocidos como informales, es decir, aquellos algoritmos (según los libros) que no pueden ser implementados a través de un computador. Yo sería un poco menos drástico. Yo diría que son algoritmos informales aquellos que no son fácilmente implementables en un computador. Segundo, y precisamente debido a que son algoritmos informales, deben hacerse una cantidad de reflexiones antes y después de ellos. Reflexiones que tienen una connotación puramente humana.

2.3. La Prueba de escritorio

En la reflexión e. se habló de una prueba. Textualmente dice: "*Si probamos este ejemplo...*", lo cual significa que todo algoritmo debe ser probado antes de ser ejecutado con el propósito de que tengamos una alta certeza en cuanto al logro del objetivo. Precisamente este es el tercer concepto que abordaremos y que se conoce como prueba de escritorio.

¿Qué es la prueba de escritorio? Es la simulación de la puesta en marcha de un algoritmo. Con la prueba de escritorio podemos determinar si el algoritmo que hemos diseñado logra el objetivo propuesto. De no ser así, podremos concluir que se debe corregir el algoritmo hasta lograr que satisfaga el objetivo propuesto.

Por lo que usted ha podido observar en el algoritmo de ejemplo, cada línea numerada del algoritmo puede considerarse a su vez como otro algoritmo, ya que

el solo hecho de saber cuál es el libro que se quiere adquirir nos obliga a realizar una serie de pasos ordenados y secuenciales para poderlo lograr. Entonces, surge una inquietud: ¿qué tan detallado debe ser un algoritmo? Su respuesta, como todo lo que va a encontrar en este libro, es muy lógica y muy sencilla.

Un algoritmo debe tener el nivel de detalle suficiente como para que no exista ninguna duda en su puesta en marcha, es decir, como para que cada línea pueda ser realizada sin el más mínimo asomo de inquietud. Esto quiere decir que algunos algoritmos pueden ser más entendibles para unas personas que para otras, dada su misma definición racional.

Como todo dentro del conocimiento humano requiere una clasificación y los conceptos de los cuales estamos hablando no son la excepción, los algoritmos se podrían clasificar en las categorías que se explican a continuación.

2.4. Algoritmos informales

Definidos como todos aquellos algoritmos que no son realizables a través de un computador (al menos, no fácilmente). Son aquellos algoritmos en donde el ejecutor real es el ser humano, como el algoritmo para dar un beso, el algoritmo para fritar unos huevos o el algoritmo para conseguir un libro. Escribo que *"…al menos no fácilmente*" porque la tecnología ha avanzado tanto que muchos algoritmos que en el pasado no eran implementables a través de un computador en la actualidad lo son y de manera mucho más sencilla, como es el caso del algoritmo para conseguir un libro que anteriormente se pensaba en librerías y ahora se piensa en un concepto más globalizado: Internet con sus buscadores, con más posibilidad de conseguirlo y con menos trabajo. De manera que vamos a considerar aquellos algoritmos informales como los que son preferiblemente realizables por el ser humano.

2.5. Algoritmos computacionales

Se consideran como tales todos aquellos algoritmos que deben ser preferiblemente implementados en un computador para aprovechar su velocidad de procesamiento. Un ejemplo de estos puede ser el algoritmo que genere los primeros 100 números primos, recordando que un número primo es aquel que solo puede ser dividido exactamente entre la unidad y entre sí mismo que, si bien podrían ser calculados utilizando un papel y un lápiz, la utilización de un computador en unión con el algoritmo adecuado nos va a dar un resultado mucho más rápido y absolutamente confiable (lo cual depende de que el programa se base en un algoritmo confiable). Son precisamente estos algoritmos los que vamos a tratar de definir y poner en práctica en el desarrollo de este libro.

En el desarrollo de los algoritmos computacionales, los cuales nos van a ocupar en lo sucesivo, la metodología para llegar a la solución final que permita lograr un objetivo (igualmente computacional) continúa con los siguientes pasos:

2.5.1. Transcripción

Este es el proceso a través del cual "convertimos" un algoritmo, escrito en términos muy coloquiales e informales, en un listado de instrucciones entendibles a un computador y que se ajustan a las reglas sintácticas de determinado lenguaje de programación. Podríamos decir que es la "traducción" de un algoritmo con la "ortografía" de un lenguaje de programación.

¿Qué son las reglas sintácticas de un lenguaje de programación? Son todas las restricciones técnicas (y algunas veces caprichosas) sobre las cuales está construido el lenguaje. Por ejemplo, y solo con el ánimo de ilustrar lo que acabo de decir, si estamos utilizando el lenguaje de programación C, la orden para leer un dato se da con la instrucción *cin*, así como está escrito y sin ningún tipo de modificación por más mínima que esta pudiera ser. Será un error, entonces, escribir esta instrucción como *Cin* o como *cim*.

El lenguaje C solo entiende la instrucción *cin* tal como sus creadores la diseñaron. De tal forma que, para escribir un algoritmo computacional en términos entendibles a un computador, lo único que necesitamos saber son las reglas sintácticas de un lenguaje de programación cualquiera. El algoritmo escrito con dichas reglas se llamará **programa**. ¿Qué es, pues, un programa? Es un algoritmo escrito con las instrucciones, las restricciones y las reglas de un lenguaje de programación.

2.5.2. Digitación

Es el proceso a través del cual le escribimos al computador el programa que hemos acabado de escribir en papel. Para ello nos valemos de un programa llamado Editor de texto o Entorno Integrado de Desarrollo (IDE, por sus siglas en inglés, *Integrated Development Environment*) que nos permite escribir un programa y grabarlo. Visto neutralmente, un programa no es más que un texto escrito bajo la óptica de algunas reglas preestablecidas por los creadores de un lenguaje de programación.

2.5.3. Compilación

Es muy normal que al reescribir un algoritmo con las reglas sintácticas de un lenguaje de programación, es decir, al escribir un programa, omitamos algunas

reglas y se nos vayan, sin querer, algunos errores. Por ejemplo, que en alguna parte del programa abrimos un paréntesis que luego se nos olvidó cerrar. Para ello, el computador nos facilita una herramienta que revisa la sintaxis del programa, nos dice si tiene errores y, en los casos más depurados, nos dice en qué líneas del programa están los errores y hasta nos sugiere la corrección.

Entonces, ¿qué es la compilación? Es el proceso a través del cual el computador revisa que el programa que hemos digitado se ajuste a las reglas sintácticas de un determinado lenguaje de programación. ¿Quién realiza realmente el proceso llamado compilación? Lo realiza un programa llamado compilador, que es el encargado de evaluar dos tipos de errores:

a) Errores de sintaxis.- Podríamos asociar los errores de sintaxis en un lenguaje de programación con los errores de ortografía en nuestro idioma. Son aquellos errores representados en la omisión de alguna o algunas reglas sintácticas (hablando de un lenguaje de programación). Por ejemplo, es normal que algunas veces, en medio de una expresión matemática, abramos un paréntesis que luego se nos olvida cerrar. En ese caso, al momento de compilar, el compilador nos indicará precisamente ese error.

b) Errores de precaución.- Algunos compiladores nos hacen, por decirlo así, cierto tipo de recomendaciones para efectos de mejoramiento o aseguramiento de nuestros programas. Este tópico lo veremos de manera más detallada en la medida que se vayan desarrollando los temas de este libro.

¿Por qué se habla de algunos compiladores? Pues porque, dado que existen varios lenguajes de programación, cada lenguaje de programación tiene su propio compilador, o sea, su propio revisor sintáctico. Podríamos decir de nuevo (y aunque sea mal dicho sirve en este caso) que la sintaxis es a un lenguaje de programación como la ortografía es a un idioma.

¿Por qué existen varios lenguajes de programación? Esto sí obedece a dos factores: el primero es la especificidad de los lenguajes, ya que son desarrollados para que cumplan de la mejor manera ciertos objetivos (refiriéndonos al mundo de la informática) y el segundo es un factor netamente comercial, pues los lenguajes de programación son producidos por empresas fabricantes de *software*.

En un programa, los errores son de tres tipos: errores de sintaxis y errores de precaución que, como ya se dijo, son revisados por el compilador. Son los errores fáciles porque los compiladores actuales no solo le dicen a uno cuál es el error, sino que además le indican, más o menos, en donde está e incluso algunas veces le sugieren la corrección. Los errores difíciles realmente de encontrar en un programa son el tercer tipo de error y son los errores lógicos, ya que

el compilador no le va a discutir acerca de lo que usted quiere hacer y cómo quiere hacerlo.

¿…Y en dónde se detectan los errores lógicos? Pues en la prueba de escritorio. Allí y solo allí usted podrá determinar si su algoritmo está realmente bien o no, es decir, si logra o no el objetivo propuesto. Ha de tenerse especial cuidado cuando un algoritmo sea transcrito, ya que el cambio de cada línea de un algoritmo por su correspondiente instrucción en un programa a veces cambia un poco la lógica inicial si no se conocen bien las reglas sintácticas del lenguaje de programación.

2.5.4. Ejecución o puesta en marcha

Luego de que hemos realizado las correcciones pertinentes para que nuestro compilador nos reporte cero errores de sintaxis y cero errores de precaución, ya estamos en condiciones de poner a "correr" nuestro programa, o sea, en condiciones de ser ejecutado por el computador. Si lo que queríamos inicialmente (o sea, nuestro objetivo) era generar los 100 primeros números pares, entonces al momento de la ejecución deberán aparecer en pantalla los 100 primeros números pares.

2.6. Verificación de resultados

Este último paso es útil ya que, con lo que nos entregue la ejecución del programa, podremos saber si se cumplió el objetivo inicial o no. En caso de que no se haya cumplido el objetivo inicial (al llegar a este punto), podría ser por algunas de las siguientes razones:

a. No teníamos claro el objetivo y fallamos en todo el proceso.

b. No realizamos bien la Prueba de Escritorio y nos la saltamos creyendo que el algoritmo estaba bien.

c. No conocíamos bien las reglas sintácticas del lenguaje con el que pensábamos trabajar y el programa transcrito final terminó siendo una representación técnica diferente del algoritmo inicial.

Lo que sí podemos asegurar es que, si mantenemos esta metodología paso a paso y cada uno lo realizamos concienzudamente, siempre al realizar la verificación de resultados se va a satisfacer con estos el objetivo inicial.

2.7. Ejercicios propuestos sobre algoritmos informales

La única forma como uno puede realmente aprender a nadar o a tirarse desde un paracaídas es haciéndolo, por eso lo invito a que se siente pacientemente a desarrollar estos algoritmos pensados para que usted encuentre una gran coincidencia entre unos y otros a pesar de tener objetivos diferentes. Sé que surgirán muchas dudas en cuanto a algunos de ellos, pero también estoy seguro de que, si usted lee este libro detenidamente, va a despejar todas las dudas que tenga. Por eso, tome al azar cualquiera de los siguientes enunciados y siéntese a practicar y a poner a funcionar, un poquito, esa lógica humana que tan pocas veces ejercitamos.

- Desarrollar un algoritmo que permita adquirir una revista.
- Desarrollar un algoritmo que permita entrar a una casa que está con llave.
- Desarrollar un algoritmo que permita dar un beso.
- Desarrollar un algoritmo que permita empacar un regalo.
- Desarrollar un algoritmo que permita encender un vehículo.
- Desarrollar un algoritmo que permita fritar un huevo.
- Desarrollar un algoritmo que permita mirar por un telescopio.
- Desarrollar un algoritmo que permita botar la basura.
- Desarrollar un algoritmo que permita tomar un baño.
- Desarrollar un algoritmo que permita estudiar para un examen.
- Desarrollar un algoritmo que permita tocar determinada canción con un instrumento musical.
- Desarrollar un algoritmo que permita viajar en avión.
- Desarrollar un algoritmo que permita encender un bombillo.
- Desarrollar un algoritmo que permita encender una vela.
- Desarrollar un algoritmo que permita apagar una vela.
- Desarrollar un algoritmo que permita apagar un bombillo.
- Desarrollar un algoritmo que permita parquear un vehículo.
- Desarrollar un algoritmo que permita almorzar.
- Desarrollar un algoritmo que permita ir de la casa al trabajo.
- Desarrollar un algoritmo que permita colocarse una camisa.
- Desarrollar un algoritmo que permita quitarse la camisa.
- Desarrollar un algoritmo que permita escuchar un determinado disco.
- Desarrollar un algoritmo que permita abrir una ventana.
- Desarrollar un algoritmo que permita ir a la tienda a comprar algo.

- Desarrollar un algoritmo que permita tomar una fotografía.
- Desarrollar un algoritmo que permita hacer deporte.
- Desarrollar un algoritmo que permita cortarse el cabello.
- Desarrollar un algoritmo que permita hacer un avión con una hoja de papel.
- Desarrollar un algoritmo que permita manejar una bicicleta.
- Desarrollar un algoritmo que permita manejar una motocicleta.
- Desarrollar un algoritmo que permita manejar un monociclo.
- Desarrollar un algoritmo que permita maquillarse.
- Desarrollar un algoritmo que permita hacer un pastel.
- Desarrollar un algoritmo que permita hacer un almuerzo.
- Desarrollar un algoritmo que permita adquirir un pantalón.
- Desarrollar un algoritmo que permita hacer un mercado pequeño.
- Desarrollar un algoritmo que permita leer el periódico.
- Desarrollar un algoritmo que permita saludar a un amigo.
- Desarrollar un algoritmo que permita arrullar a un bebé hasta que se duerma.
- Desarrollar un algoritmo que permita hacer un gol en fútbol.
- Desarrollar un algoritmo que permita jugar ping-pong.
- Desarrollar un algoritmo que permita nadar.
- Desarrollar un algoritmo que permita tirarse desde un avión con un paracaídas.
- Desarrollar un algoritmo que permita tirarse desde un avión sin un paracaídas.
- Desarrollar un algoritmo que permita descifrar un jeroglífico.
- Desarrollar un algoritmo que permita amarrarse un zapato.
- Desarrollar un algoritmo que permita quitarse los zapatos.
- Desarrollar un algoritmo que permita silbar.
- Desarrollar un algoritmo que permita elevar una cometa.
- Desarrollar un algoritmo que permita desarrollar algoritmos.

Capítulo 3

Variables, constantes y operadores

3.1. Variable

Informalmente, algo variable es algo que puede cambiar de un momento a otro. Técnicamente, una variable es un campo de memoria al que se le puede cambiar su contenido cuantas veces sea necesario. Primera aclaración: un campo de memoria es un pedacito de la memoria principal del computador en donde podemos guardar un dato. Segunda aclaración: a pesar de que en la memoria es donde se guarda la información, exactamente esta se almacena en variables. Esto le ha de representar a usted que es, a través de variables, como se puede utilizar la memoria del computador.

¿Ha notado usted que la maleta de una guitarra es diferente a la maleta de un violín o de una trompeta? Sabe entonces ¿qué es lo que diferencia la maleta de un instrumento musical de la maleta de otro instrumento musical? Pues precisamente la única diferencia es su contenido, es decir, el instrumento en sí. Y esto, ¿qué tiene que ver con el tema que estamos tratando? Pues muy sencillo: la diferencia entre una variable y otra radica precisamente en su contenido o, más bien, en el tipo de su contenido.

Para poder utilizar variables en el desarrollo de un programa de computador se debe primero decir qué tipo de dato se va a almacenar, pues las variables son como unas cajitas de diferentes tamaños y, por tal motivo, se deben declarar previamente para que el computador las dimensione de acuerdo a las necesidades. ¿Cuáles son los tipos de datos que pueden ser almacenados en una variable? A pesar del avance de la tecnología, los tipos de datos de las variables se explican a continuación.

3.1.1. Tipo entero

Un dato de tipo entero es un número que no tiene punto decimal, por lo tanto, en sus operaciones jamás va a generar decimales. Por ejemplo 25, -96 y 0. El hecho de que los datos de tipo entero no generen decimales significa que operan con un juego de reglas llamado aritmética entera. Una variable que se declare de tipo entero podrá almacenar solamente datos de tipo entero.

3.1.2. Tipo real

Un dato de tipo real es un número que tiene punto decimal, por lo tanto, en sus operaciones puede generar decimales. Por ejemplo 12.3, -78.56 o 45.0. El hecho de que los datos de tipo real generen decimales significa que operan con un juego de reglas llamado aritmética real. Una variable que se declare de tipo real podrá almacenar solamente datos de tipo real.

Por lo dicho en las anteriores dos definiciones, ¿qué tipo de dato sería *5.*? (así, con el punto y todo). Pensaríamos que es un entero, pero en realidad no. La definición de dato entero es que no tiene punto decimal y la de dato real es que tiene punto decimal, por lo tanto, 5. es un dato real.

3.1.3. Tipo carácter

Un dato tipo carácter es un equivalente del código ASCII (*American Standard Code for Interchange Information*). ¿Qué es el código ASCII? Es el código internacional de equivalencias internas en el sistema binario. A nivel mundial, los computadores están construidos en un sistema numérico llamado sistema binario, sistema que se basa solamente en la utilización de unos (1) y ceros (0). Este sistema tiene una relación directa con el sistema decimal y, por lo tanto, fue adoptado, ya que permite aprovechar características físicas de los componentes electrónicos. Dada la gran importancia que poco a poco fueron adquiriendo los computadores, se adoptó un solo código interno para la interpretación de todas y cada una de las teclas de su teclado.

De esta forma, cuando usted presiona en su teclado la letra A, realmente se genera por dentro de su computador el número 65 pero expresado en código binario, es decir, 0100 0001, y cuando usted presiona la tecla 1, se genera internamente el número 49, pero expresado igualmente en código binario, es decir, 0011 0001.

Cada una de las teclas que usted presione tendrá un equivalente interno y por supuesto expresado (internamente) en sistema binario. Cada cero o cada uno

utilizado en este sistema se conoce como bit (abreviatura de *binary digit*) y un conjunto de 8 bits (medida en la cual se expresa el código ASCII) se conoce como un byte (pronúnciese *bait*).

Como el código ASCII está expresado en bytes y cada byte tiene 8 bits y cada bit puede tener un 0 o un 1 (o sea, dos estados), entonces se puede concluir que el código completo consta de 2^8 combinaciones (o sea, 256 equivalencias). A continuación relaciono la tabla completa de equivalencias ASCII.

Tabla 5. Fragmento del código ASCII

CÓDIGO ASCII					
Carácter	**Equivalencia sistema decimal**	**Equivalencia sistema binario**	**Carácter**	**Equivalencia sistema decimal**	**Equivalencia sistema binario**
0	48	0011 0000	G	71	0100 0111
1	49	0011 0001	H	72	0100 1000
2	50	0011 0010	I	73	0100 1001
3	51	0011 0011	J	74	0100 1010
4	52	0011 0100	a	97	0110 0001
5	53	0011 0101	b	98	0110 0010
6	54	0011 0110	c	99	0110 0011
7	55	0011 0111	d	100	0110 0100
8	56	0011 1000	e	101	0110 0101
9	57	0011 1001	f	102	0110 0110
A	65	0100 0001	g	103	0110 0111
B	66	0100 0010	h	104	0110 1000
C	67	0100 0011	i	105	0110 1001
D	68	0100 0100	j	106	0110 1010

Como puede usted notar, estas son apenas algunas de las 256 equivalencias que tiene la tabla ASCII. Es obvio pensar que también tienen equivalencia los caracteres especiales como la coma, el punto o el paréntesis.

Cuando se tiene un conjunto de caracteres, se dice técnicamente que se tiene una cadena, por lo tanto, el nombre del autor "OMAR" es una cadena. El contenido de una cadena no es evaluado por el computador y se acostumbra acotarlo o encerrarlo entre comillas dobles; así, la cadena "5 – 7 es igual a 8", a pesar de no ser lógica ni correcta matemáticamente, es válida para el computador ya que él, en ningún momento, evalúa las cadenas.

3.2. Asignaciones

¿Cómo se llevan los datos a las variables?, o sea, ¿cómo se "cargan" las variables? Pues a través de un signo muy conocido por usted y es el signo =.

Este signo tiene, en el caso de los algoritmos computacionales y programas, una connotación un poco diferente a la que se le da en matemáticas. El signo igual (=) significa que el computador va a realizar lo que está a la derecha del igual y lo va a almacenar en la variable que se encuentre a la izquierda del igual.

De manera que ya usted puede ver claramente en esta definición que a la izquierda del igual solo puede haber una variable y al lado derecho del igual puede haber una constante, una variable o una expresión. De manera que cualquiera de los siguientes esquemas es válido:

a = 8 Le indica al computador que guarde la constante 8 en la variable a.

b = a Le indica al computador que guarde en la variable b el contenido de la variable a que en la instrucción había sido "cargada" con 8, por lo tanto, en la variable b queda el valor de 8 al igual que en la variable a.

c = a + bLe indica al computador que guarde en la variable c el resultado de sumar el contenido de la variable a con el contenido de la variable b. Como la variable a tenía el contenido 8 y la variable b también tenía el contenido 8, entonces el computador sumará 8+8 y ese 16 de resultado lo almacenará en la variable c.

Puede notarse en este ejemplo que en la variable a se ha almacenado una constante, en la variable b se ha almacenado el contenido de otra variable y en la variable c se ha almacenado el resultado de una expresión. Y qué pasará si luego de tener estas tres instrucciones adicionamos la siguiente:

b = 9 Pues, muy sencillo; el anterior contenido de la variable b que era 8 va a ser reemplazado por el nuevo contenido de la variable b que es 9. Esto significa que, cada vez que se asigna un nuevo valor (proveniente de una constante, una variable o como resultado de una expresión), el valor anterior de la misma variable se pierde.

De esta forma, si se quisieran escribir los contenidos de las variables a, b y c, el computador nos reportaría para a el contenido 8, para b el contenido 9 y para c el contenido 16.

Todo lo que debe tener en cuenta con la asignación o carga de las variables es lo siguiente:

a. Al lado izquierdo del igual solo puede haber una variable.

b. Al lado derecho del igual puede haber una constante, una variable o una expresión.

c. El computador siempre resuelve lo de la derecha del igual y su resultado lo almacena en la variable que esté a la izquierda del mismo.

d. Cada vez que se le entra un nuevo valor a una variable, el valor anterior se pierde.

De acuerdo a lo dicho, vamos a resolver el siguiente conjunto de instrucciones:

Entero: A, B, C	Declara de tipo entero las variables A, B y C, de manera que solo podrán almacenar datos enteros.
A = 10	Almacena la constante 10 en la variable A.
B = 15	Almacena la constante 15 en la variable B.
C = 20	Almacena la constante 20 en la variable C.
A = A + B	Almacena en la variable A el resultado de sumar el contenido de A más el contenido de B, o sea, 10+15, que es igual a 25.
B = B + 8	Almacena en la variable B el resultado de sumar el contenido de B con la constante 8, o sea, 15+8, que es igual a 23.
C = C + A	Almacena en la variable C el resultado de sumar el contenido de la variable C más el contenido de la variable A, o sea, 20+25, que es igual a 45. Recuerde que en esta línea se utiliza el último valor almacenado en la variable A.
A = A + 5	Almacena en la variable C el resultado de sumar el contenido de la variable A más la constante 5, es decir, 25+5, que es igual a 30.
B = B + 3	Almacena en la variable B el resultado de sumar el contenido de la variable B más la constante 3, o sea, 23+3, que es igual a 26.
C = C + 2	Almacena en la variable C el resultado de sumar el contenido de la variable C más la constante 2, o sea, 45+2, que es igual a 47.
A = A – B	Almacena en la variable A el resultado de restarle al contenido de la variable A el contenido de la variable B, o sea, 30-26, que es igual a 4.
B = A – B	Almacena en la variable B el resultado de restarle al contenido de la variable A el contenido de la variable B, o sea, 4-26, que es igual a -22.

C = A – B	Almacena en la variable C el resultado de restarle al contenido de la variable A el contenido de la variable B, o sea, 4- (-22), que por propiedades algebraicas es igual a 4+22, o sea, 26.

Los resultados finales en las tres variables son:

Variable A	4
Variable B	-22
Variable C	26

No olvide que para el manejo de variables cada nuevo valor que se le asigne a una variable borra el valor anterior. Nótese que en este conjunto de instrucciones las tres últimas son iguales en su forma pero no en sus resultados. Para hacerlo más breve, el seguimiento de este conjunto de instrucciones podríamos detallarlo de la siguiente forma:

Variables

Entero: A, B, C	**A**	**B**	**C**
A = 10	10		
B = 15	10	15	
C = 20	~~10~~	15	20
A = A + B	25	~~15~~	20
B = B + 8	25	23	~~20~~
C = C + A	~~25~~	23	45
A = A + 5	30	~~23~~	45
B = B + 3	30	26	~~45~~
C = C + 2	~~30~~	26	47
A = A – B	4	~~26~~	47
B = A – B	4	-22	~~47~~
C = A – B	4	-22	26

Era evidente que teníamos que llegar al mismo resultado. Esto que acabamos de hacer es precisamente la PRUEBA DE ESCRITORIO de este conjunto de instrucciones. También puede notarse que cada nuevo valor asignado a cada variable reemplaza el valor anterior de la misma variable, por esa razón, por cada nuevo resultado (en una determinada variable), se va tachando el resultado anterior para indicar que ese ya no es válido.

3.3. Ejercicios

1.

```
a = 10
b = 20
c = 5
a = a + 3
b = b + 4 – a
c = a + b + c
a = a + c
b = 4
c = c + 3 - b + 2
```

¿Qué valores quedan almacenados en las variables a, b y c?

2.

```
a = 5
b = 18
c = 15
d = 25
a = a + 10
b = b + 5 – c
c = c + 4 + b
d = d + b + a
a = a + 1
b = b + c
c = b + c
d = b + b
```

¿Qué valores quedan almacenados en las variables a, b, c y d?

3.

```
a = 9
b = 6
a = a + 4
b = b + 2
a = a + 10
```