

# PROBLEMAS, SOLUCIONES Y PROGRAMAS

01



# 1. Objetivos Pedagógicos

Al final de este nivel el lector será capaz de:

- Explicar el proceso global de solución de un problema usando un [programa de computador](#). Esto incluye las etapas que debe seguir para resolverlo y los distintos elementos que debe ir produciendo a medida que construye la solución.
- Analizar un problema simple que se va a resolver usando un [programa de computador](#), construyendo un modelo con los elementos que intervienen en el problema y especificando los servicios que el programa debe ofrecer.
- Explicar la estructura de un [programa de computador](#) y el papel que desempeña cada uno de los elementos que lo componen.
- Completar una solución parcial a un problema (un programa incompleto escrito en el lenguaje Java), usando expresiones simples, asignaciones e invocaciones a métodos. Esto implica entender los conceptos de [parámetro](#) y de creación de objetos.
- Utilizar un [ambiente de desarrollo](#) de programas y un espacio de trabajo predefinido, para completar una solución parcial a un problema.

## 2. Motivación

La computación es una disciplina joven comparada con las matemáticas, la física o la ingeniería civil. A pesar de su juventud, nuestra vida moderna depende de los computadores. Desde la nevera de la casa, hasta el automóvil y el teléfono celular, todos requieren de programas de computador para funcionar. Se ha preguntado alguna vez, ¿cuántas líneas de código tienen los programas que permiten volar a un avión? La respuesta es varios millones.

El computador es una herramienta de trabajo, que nos permite aumentar nuestra productividad y tener acceso a grandes volúmenes de información. Es así como, con un computador, podemos escribir documentos, consultar los horarios de cine, bajar música de Internet, jugar o ver películas. Pero aún más importante que el uso personal que le podemos dar a un computador, es el uso que hacen de él otras disciplinas. Sería imposible sin los computadores llegar al nivel de desarrollo en el que nos encontramos en disciplinas como la biología (¿qué sería del estudio del genoma sin el computador?), la medicina, la ingeniería mecánica o la aeronáutica. El computador nos ayuda a almacenar grandes cantidades de información (por ejemplo, los tres mil millones de pares de bases del genoma humano, o los millones de píxeles que conforman una imagen que llega desde un satélite) y a manipularla a altas velocidades, para poder así ejecutar tareas que hasta hace sólo algunos años eran imposibles para nosotros.

El usuario de un [programa de computador](#) es aquél que, como parte de su trabajo o de su vida personal, utiliza las aplicaciones desarrolladas por otros para resolver un problema. Todos nosotros somos usuarios de editores de documentos o de navegadores de Internet, y los usamos como herramientas para resolver problemas. Un programador, por su parte, es la persona que es capaz de entender los problemas y necesidades de un usuario y, a partir de dicho conocimiento, es capaz de construir un [programa de computador](#) que los resuelva (o los ayude a resolver). Vista de esta manera, la programación se puede considerar fundamentalmente una actividad de servicio para otras disciplinas, cuyo objetivo es ayudar a resolver problemas, construyendo soluciones que utilizan como herramienta un computador.

Cuando el problema es grande (como el sistema de información de una empresa), complejo (como crear una visualización tridimensional de un [diseño](#)) o crítico (como controlar un tren), la solución la construyen equipos de ingenieros de software, entrenados especialmente para asumir un reto de esa magnitud. En ese caso aparecen también los arquitectos de software, capaces de proponer una estructura adecuada para conectar los componentes del programa, y un conjunto de expertos en redes, en bases de datos, en el

negocio de la compañía, en **diseño** de interfaces gráficas, etc. Cuanto más grande es el problema, más interdisciplinaridad se requiere. Piense que en un proyecto grande, puede haber más de 1000 expertos trabajando al mismo tiempo en el **diseño** y construcción de un programa, y que ese programa puede valer varios miles de millones de dólares. No en vano, la industria de construcción de software mueve billones de dólares al año.

Independiente del tamaño de los programas, podemos afirmar que la programación es una actividad orientada a la solución de problemas. De allí surgen algunos de los interrogantes que serán resueltos a lo largo de este primer nivel: ¿Cómo se define un problema? ¿Cómo, a partir del problema, se construye un programa para resolverlo? ¿De qué está conformado un programa? ¿Cómo se construyen sus partes? ¿Cómo se hace para que el computador entienda la solución?

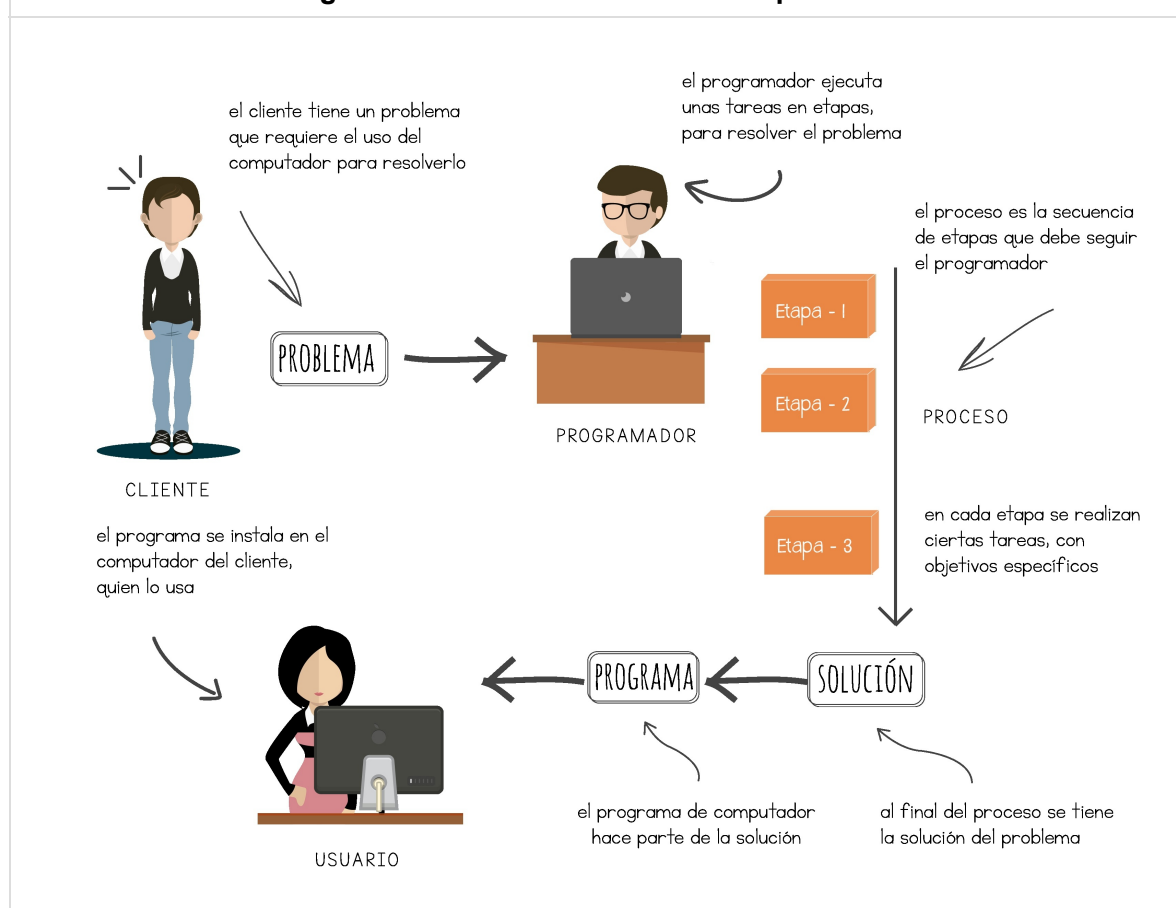
Bienvenidos, entonces, al mundo de la construcción de programas. Un mundo en **constante** evolución, en donde hay innumerables áreas de aplicación y posibilidades profesionales.

### 3. Problemas y Soluciones

Sigamos el escenario planteado en la [figura 1.1](#), el cual resume el ciclo de vida de construcción de un programa y nos va a permitir introducir la terminología básica que necesitamos:

- **Paso 1:** Una persona u organización, denominada el **cliente**, tiene un problema y necesita la construcción de un programa para resolverlo. Para esto contacta una empresa de desarrollo de software que pone a su disposición un **programador**.
- **Paso 2:** El programador sigue un conjunto de etapas, denominadas el **proceso**, para entender el problema del cliente y construir de manera organizada una **solución** de buena calidad, de la cual formará parte un **programa**.
- **Paso 3:** El programador instala el programa que resuelve el problema en un computador y deja que el **usuario** lo utilice para resolver el problema. Fíjese que no es necesario que el cliente y el usuario sean la misma persona. Piense por ejemplo que el cliente puede ser el gerente de producción de una fábrica y, el usuario, un operario de la misma.

**Fig. 1.1 Proceso de solución de un problema**



- En la primera sección nos concentramos en la definición del problema, en la segunda en el proceso de construcción de la solución y, en la tercera, en el contenido y estructura de la solución misma.

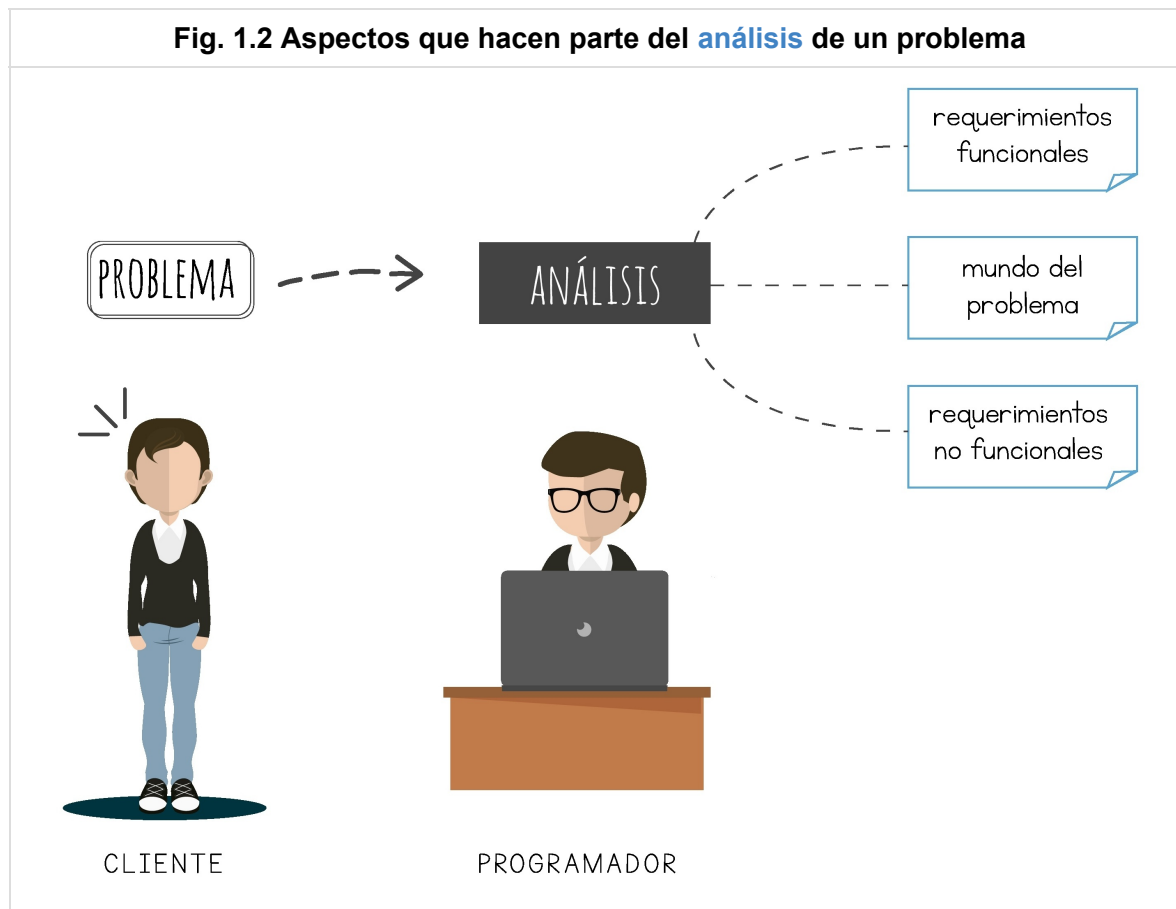
## 3.1. Especificación de un Problema

Partimos del hecho de que un programador no puede resolver un problema que no entiende. Por esta razón, la primera etapa en todo proceso de construcción de software consiste en tratar de entender el problema que tiene el cliente, y expresar toda la información que él suministre, de manera tal que cualquier otra persona del equipo de desarrollo pueda entender sin dificultad lo que espera el cliente de la solución. Esta etapa se denomina **análisis** y la salida de esta etapa la llamamos la **especificación** del problema.

Para introducir los elementos de la **especificación**, vamos a hacer el paralelo con otras ingenierías, que comparten problemáticas similares. Considere el caso de un ingeniero civil que se enfrenta al problema de construir una carretera. Lo primero que éste debe hacer es tratar de entender y especificar el problema que le plantean. Para eso debe tratar de identificar al menos tres aspectos del problema: (1) los requerimientos del usuario (entre qué puntos quiere el cliente la carretera, cuántos carriles debe tener, para qué tipo de tráfico debe ser la carretera), (2) el mundo en el que debe resolverse el problema (el tipo de terreno, la cantidad de lluvia, la temperatura), y (3) las restricciones y condiciones que plantea el cliente (el presupuesto máximo, que las pendientes no sobrepasen el 5%). Sería una pérdida de tiempo y de recursos para el ingeniero civil, intentar construir la carretera si no ha entendido y definido claramente los tres puntos antes mencionados. Y más que tiempo y recursos, habrá perdido algo muy importante en una profesión de servicio como es la ingeniería, que es la confianza del cliente.

En general, todos los problemas se pueden dividir en estos tres aspectos. Por una parte, se debe identificar lo que el cliente espera de la solución. Esto se denomina un **requerimiento funcional**. En el caso de la programación, un **requerimiento funcional** hace referencia a un servicio que el programa debe proveer al usuario. El segundo aspecto que conforma un problema es el **mundo** o **contexto** en el que ocurre el problema. Si alguien va a escribir un programa para una empresa, no le basta con entender la funcionalidad que éste debe tener, sino que debe entender algunas cosas de la estructura y funcionamiento de la empresa. Por ejemplo, si hay un **requerimiento funcional** de calcular el salario de un empleado, la descripción del problema debe incluir las normas de la empresa para calcular un salario. El tercer aspecto que hay que considerar al definir un problema son los **requerimientos no funcionales**, que corresponden a las restricciones o condiciones que impone el cliente al programa que se le va a construir. Fíjese que estos últimos se utilizan para limitar las

soluciones posibles. En el caso del programa de una empresa, una restricción podría ser el tiempo de entrega del programa, o la cantidad de usuarios simultáneos que lo deben poder utilizar. En la [figura 1.2](#) se resumen los tres aspectos que conforman un problema.



- Analizar un problema es tratar de entenderlo. Esta etapa busca garantizar que no tratemos de resolver un problema diferente al que tiene el cliente.
- Descomponer el problema en sus tres aspectos fundamentales, facilita la tarea de entenderlo: en cada etapa nos podemos concentrar en sólo uno de ellos, lo cual simplifica el trabajo.
- Esta descomposición se puede generalizar para estudiar todo tipo de problemas, no sólo se utiliza en problemas cuya solución sea un [programa de computador](#).
- Además de entender el problema, debemos expresar lo que entendemos siguiendo algunas convenciones.
- Al terminar la etapa de [análisis](#) debemos generar un conjunto de documentos que contendrán nuestra comprensión del problema. Con dichos documentos podemos validar nuestro trabajo, presentándoselo al cliente y discutiendo con él.

## Ejemplo 1

**Objetivo:** Identificar los aspectos que hacen parte de un problema.

El problema: una empresa de aviación quiere construir un programa que le permita buscar una ruta para ir de una ciudad a otra, usando únicamente los vuelos de los que dispone la empresa. Se quiere utilizar este programa desde todas las agencias de viaje del país.

Cliente	La empresa de aviación.
Usuario	Las agencias de viaje del país.
Requerimiento funcional	R1: dadas dos ciudades C1 y C2, el programa debe dar el itinerario para ir de C1 a C2, usando los vuelos de la empresa. En este ejemplo sólo hay un <b>requerimiento funcional</b> explícito. Sin embargo, lo usual es que en un problema haya varios de ellos.
Mundo del problema	En el enunciado no está explícito, pero para poder resolver el problema, es necesario conocer todos los vuelos de la empresa y la lista de ciudades a las cuales va. De cada vuelo es necesario tener la ciudad de la que parte, la ciudad a la que llega, la hora de salida y la duración del vuelo. Aquí debe ir todo el conocimiento que tenga la empresa que pueda ser necesario para resolver los requerimientos funcionales.
Requerimiento no funcional	El único <b>requerimiento no funcional</b> mencionado en el enunciado es el de distribución, ya que las agencias de viaje están geográficamente dispersas y se debe tener en cuenta esta característica al momento de construir el programa.

## Tarea 1:

**Objetivo:** Identificar los aspectos que forman parte de un problema.

El problema: un banco quiere crear un programa para manejar sus cajeros automáticos. Dicho programa sólo debe permitir retirar dinero y consultar el saldo de una cuenta. Identifique y discuta los aspectos que constituyen el problema. Si el enunciado no es explícito con respecto a algún punto, intente imaginar la manera de completarlo.



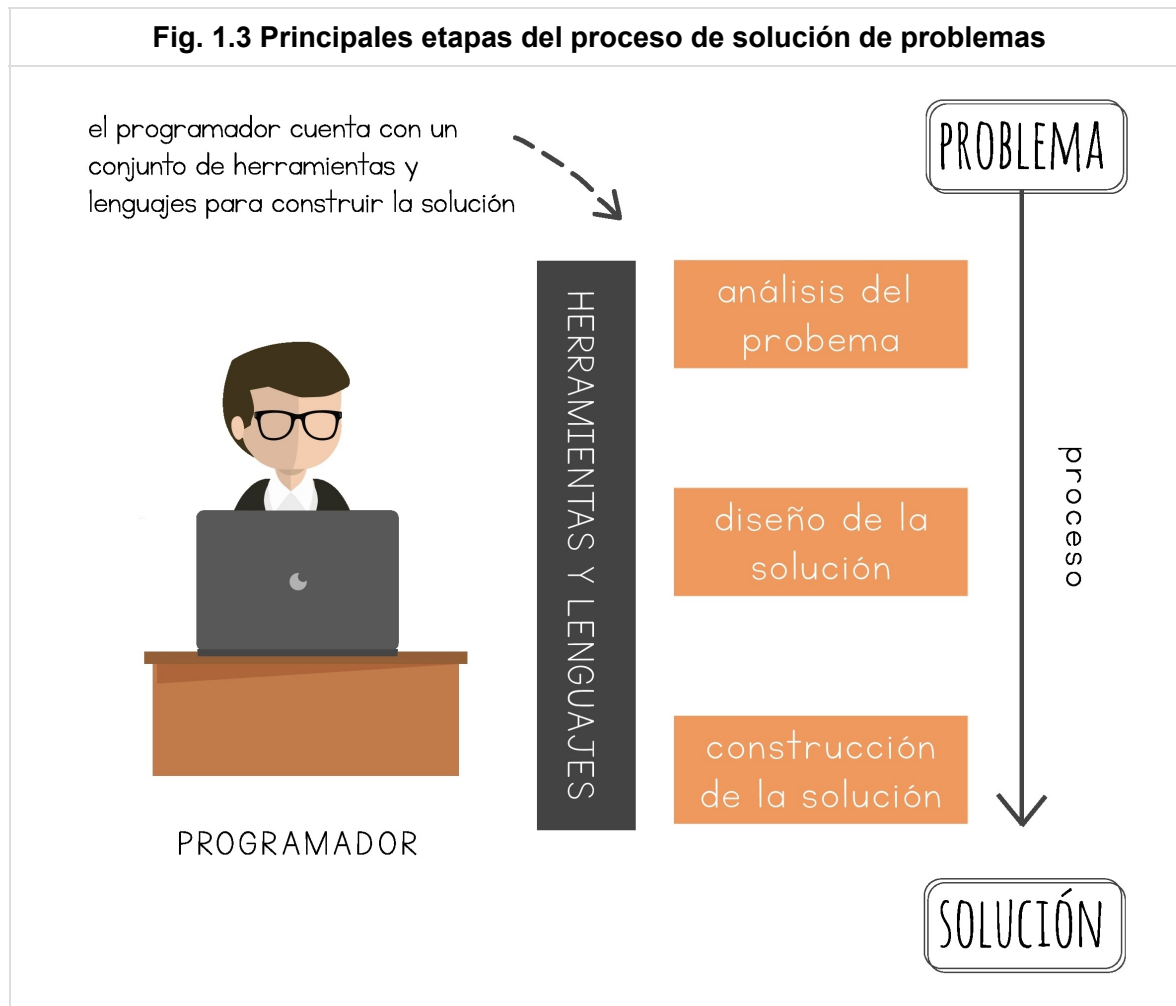
<b>Cliente</b>	
Usuario	
Requerimiento funcional	
Mundo del problema	
Requerimiento no funcional	

Analizar un problema significa entenderlo e identificar los tres aspectos en los cuales siempre se puede descomponer: los requerimientos funcionales, el mundo del problema y los requerimientos no funcionales. Esta división es válida para problemas de cualquier tamaño.

## 3.2. El Proceso y las Herramientas

Entender y especificar el problema que se quiere resolver es sólo la primera etapa dentro del proceso de desarrollo de un programa. En la [figura 1.3](#) se hace un resumen de las principales etapas que constituyen el proceso de solución de un problema. Es importante que el lector entienda que si el problema no es pequeño (por ejemplo, el sistema de información de una empresa), o si los requerimientos no funcionales son críticos (por ejemplo, el sistema va a ser utilizado simultáneamente por cincuenta mil usuarios), o si el desarrollo se hace en equipo (por ejemplo, veinte ingenieros trabajando al mismo tiempo), es necesario adaptar las etapas y la manera de trabajar que se plantean en este libro. En este libro sólo abordamos la problemática de construcción de programas de computador para resolver problemas pequeños.

**Fig. 1.3 Principales etapas del proceso de solución de problemas**



- La primera etapa para resolver un problema es analizarlo. Para facilitar este estudio, se debe descomponer el problema en sus tres partes.
- Una vez que el problema se ha entendido y se ha expresado en un lenguaje que se pueda entender sin ambigüedad, pasamos a la etapa de **diseño**. Aquí debemos imaginarnos la solución y definir las partes que la van a componer. Es muy común comenzar esta etapa definiendo una estrategia.
- Cuando el **diseño** está terminado, pasamos a construir la solución.

El proceso debe ser entendido como un orden en el cual se debe desarrollar una serie de actividades que van a permitir construir un programa. El proceso planteado tiene tres etapas principales, todas ellas apoyadas por herramientas y lenguajes especiales:

- **Análisis del problema:** el objetivo de esta etapa es entender y especificar el problema que se quiere resolver. Al terminar, deben estar especificados los requerimientos funcionales, debe estar establecida la información del mundo del problema y deben estar definidos los requerimientos no funcionales.
- **Diseño de la solución:** el objetivo es detallar, usando algún lenguaje (planos, dibujos, ecuaciones, diagramas, texto, etc.), las características que tendrá la solución antes de ser construida. Los diseños nos van a permitir mostrar la solución antes de comenzar el proceso de fabricación propiamente dicho. Es importante destacar que dicha **especificación** es parte integral de la solución.
- **Construcción de la solución:** tiene como objetivo **implementar** el programa a partir del **diseño** y probar su correcto funcionamiento.

Cada una de las etapas de desarrollo está apoyada por herramientas y lenguajes, que van a permitir al programador expresar el producto de su trabajo. En la etapa de construcción de la solución es conveniente contar con un **ambiente de desarrollo** que ayuda, entre otras cosas, a editar los programas y a encontrar los errores de sintaxis que puedan existir.

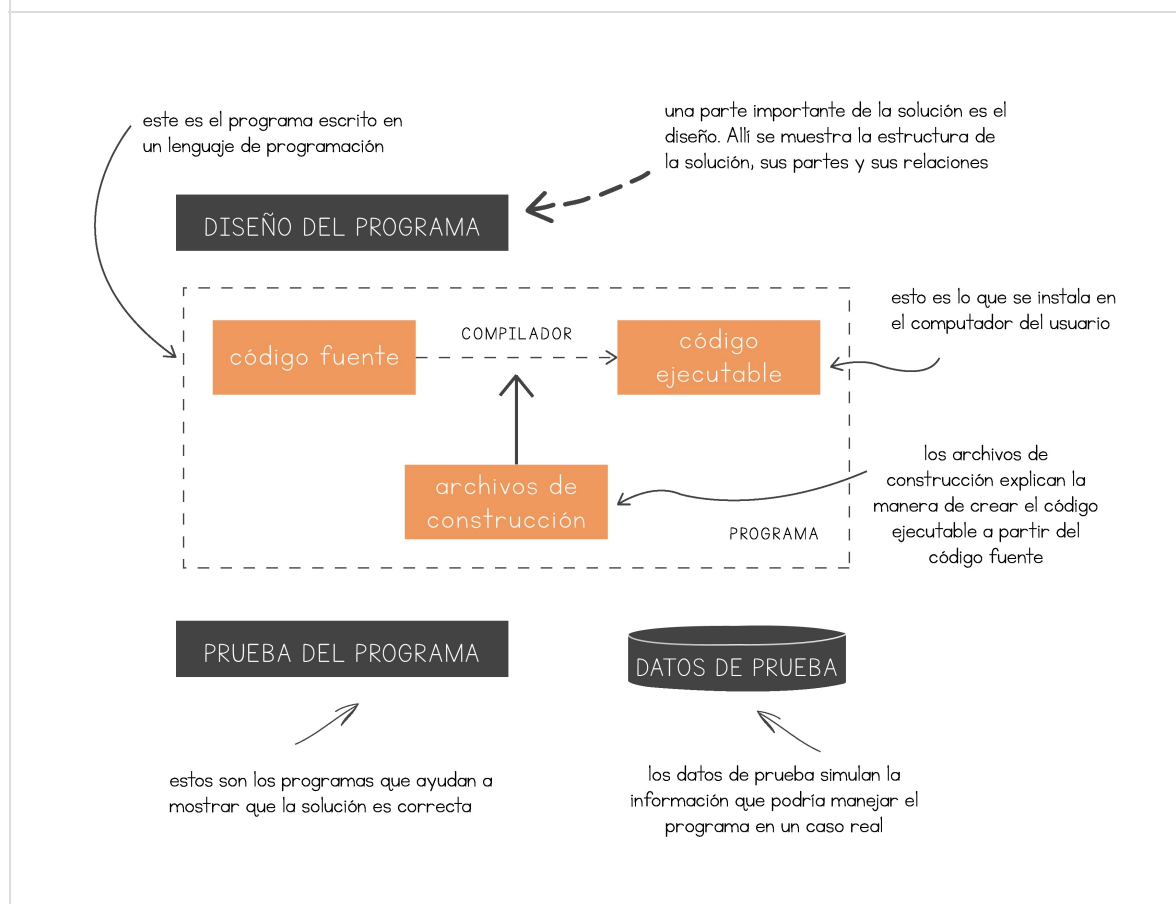
Las etapas iniciales del proceso de construcción de un programa son críticas, puesto que cuanto más tarde se detecta un error, más costoso es corregirlo. Un error en la etapa de **análisis** (entender mal algún aspecto del problema) puede implicar la pérdida de mucho tiempo y dinero en un proyecto. Es importante que al finalizar cada etapa, tratemos de asegurarnos de que vamos avanzando correctamente en la construcción de la solución.

### 3.3. La Solución a un Problema

La solución a un problema tiene varios componentes, los cuales se ilustran en la **figura 1.4**. El primero es el **diseño** (los planos de la solución) que debe definir la estructura del programa y facilitar su posterior mantenimiento. El segundo elemento es el **código fuente**

del programa, escrito en algún **lenguaje de programación** como Java, C, C# o C++. El **código fuente** de un programa se crea y edita usando el **ambiente de desarrollo** mencionado en la sección anterior.

**Fig. 1.4 Elementos que forman parte de la solución de un problema**



Existen muchos tipos de lenguajes de programación, entre los cuales los más utilizados en la actualidad son los llamados lenguajes de **programación orientada a objetos**. En este libro utilizaremos Java que es un lenguaje orientado a objetos muy difundido y que iremos presentando poco a poco, a medida que vayamos necesitando sus elementos para resolver problemas.

Un programa es la secuencia de instrucciones (escritas en un **lenguaje de programación**) que debe ejecutar un computador para resolver un problema.

El tercer elemento de la solución son los archivos de construcción del programa. En ellos se explica la manera de utilizar el **código fuente** para crear el **código ejecutable**. Este último es el que se instala y ejecuta en el computador del usuario. El programa que permite traducir el **código fuente** en **código ejecutable** se denomina **compilador**. Antes de poder construir nuestro primer programa en Java, por ejemplo, tendremos que conseguir el respectivo **compilador** del lenguaje.

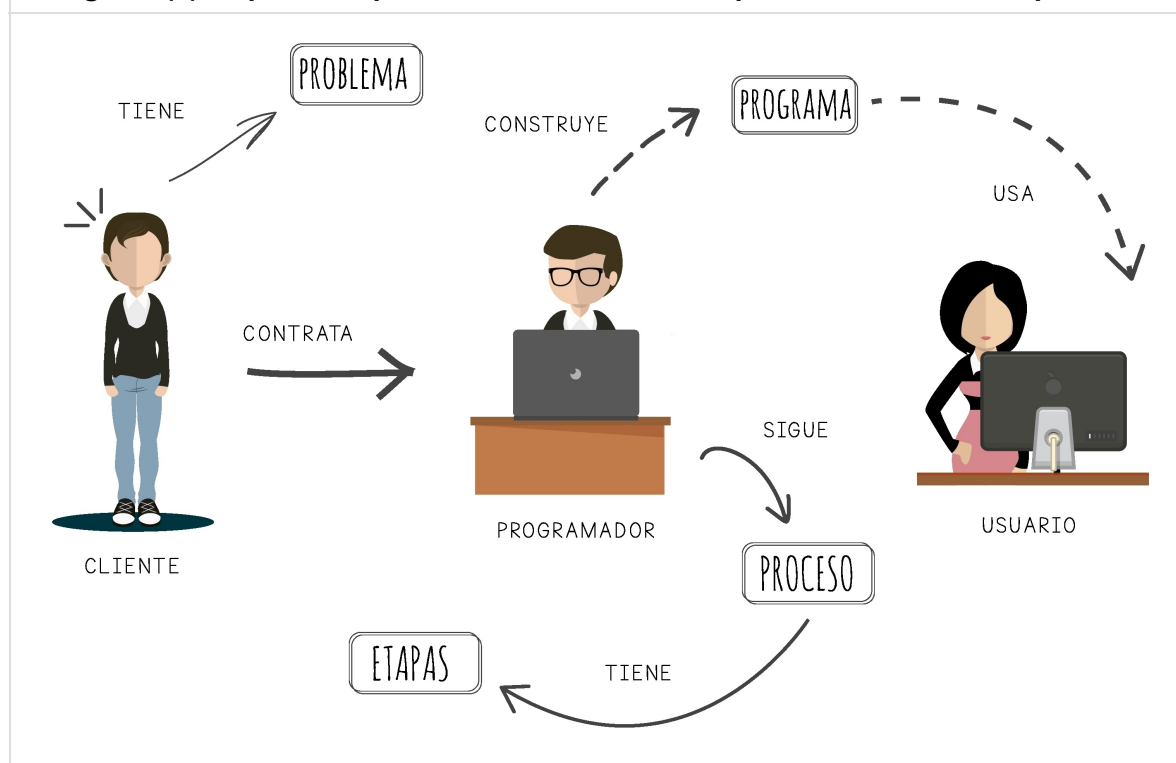
El último elemento que forma parte de la solución son las **pruebas**. Allí se tiene un programa que es capaz de probar que el programa que fue entregado al cliente funciona correctamente. Dicho programa funciona sobre un conjunto predefinido de datos, y es capaz de validar que para esos datos predefinidos (y que simulan datos reales), el programa funciona bien.

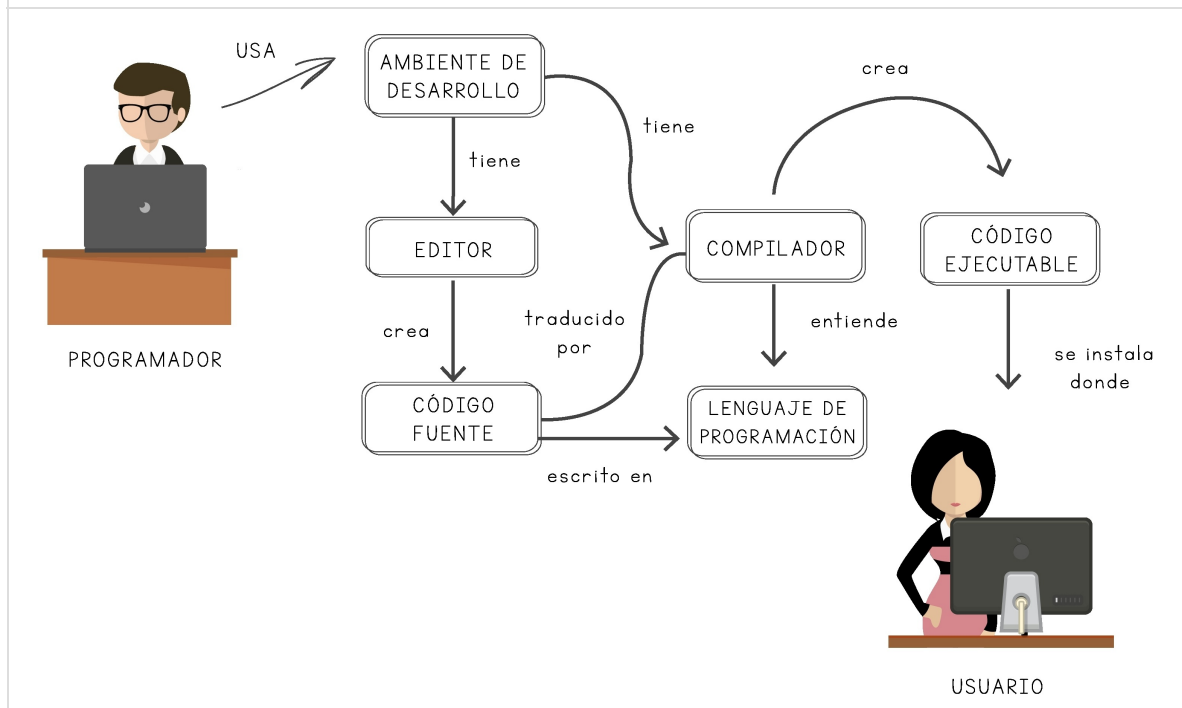
La solución de un problema tiene tres partes: (1) el **diseño**, (2) el programa y (3) las pruebas de corrección del programa. Estos son los elementos que se deben entregar al cliente. Es común que, además de los tres elementos citados anteriormente, la solución incluya un manual del usuario, que explique el funcionamiento del programa.

Si por alguna razón el problema del cliente evoluciona (por ejemplo, si el cliente pide un nuevo **requerimiento funcional**), cualquier programador debe poder leer y entender el **diseño**, añadirle la modificación pedida, ajustar el programa y extender las pruebas para verificar la nueva extensión.

La figura 1.5 muestra dos mapas conceptuales (**parte a** y **parte b**) que intentan resumir lo visto hasta el momento en este capítulo.

**Fig. 1.5 (a) Mapa conceptual de la solución de un problema con un computador**



**Fig. 1.5 (b) Mapa conceptual de la construcción de un programa**

## 4. Casos de Estudio

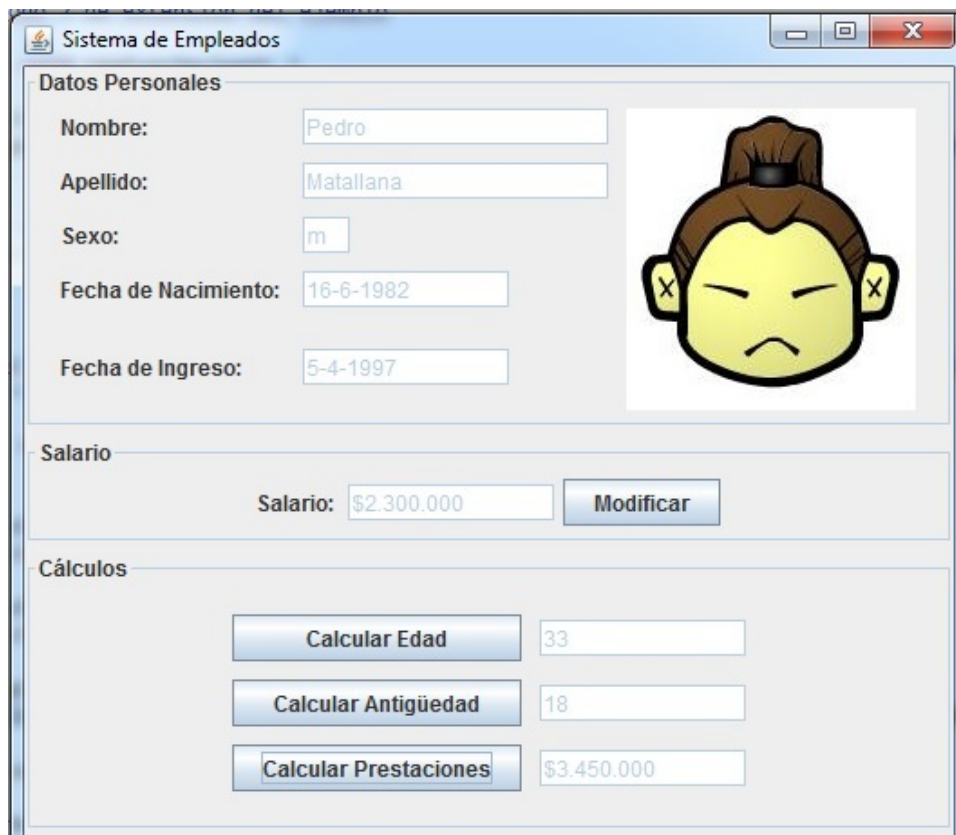
Los tres casos de estudio que se presentan a continuación serán utilizados en el resto del capítulo para ilustrar los conceptos que se vayan introduciendo. Se recomienda leerlos detenidamente antes de continuar y tratar de imaginar el funcionamiento de los programas que resuelven los problemas, utilizando para esto las figuras que se muestran. Al final del capítulo encontrará otros casos de estudio diferentes, con las respectivas hojas de trabajo para desarrollarlos.

### 4.1 Caso de Estudio N° 1: Un Empleado

Para este caso de estudio vamos a considerar un programa que administra la información de un empleado.

El empleado tiene un nombre, un apellido, un género (masculino o femenino), una fecha de nacimiento y una imagen asociada (su foto). Además, tiene una fecha de ingreso a la empresa en la que trabaja y un salario básico asignado.

Desde el programa se debe poder cambiar el salario del empleado, lo mismo que hacer los siguientes cálculos con la información disponible: (1) edad actual, (2) antigüedad en la empresa y (3) prestaciones a las que tiene derecho en la empresa.



**Sistema de Empleados**

**Datos Personales**

Nombre:

Apellido:

Sexo:

Fecha de Nacimiento:

Fecha de Ingreso:

**Salario**

Salario:

**Cálculos**

## 4.2. Caso de Estudio N° 2: Un Simulador Bancario

Una de las actividades más comunes en el mundo financiero es la realización de simulaciones que permitan a los clientes saber el rendimiento de sus productos a través del tiempo, contemplando diferentes escenarios y posibles situaciones que se presenten.

Se quiere crear un programa que haga la simulación en el tiempo de la cuenta bancaria de un cliente. Un cliente tiene un nombre y un número de cédula el cual identifica la cuenta. Una cuenta, por su parte, está constituida por tres productos financieros básicos: (1) Una cuenta de ahorro, (2) una cuenta corriente y (3) un certificado de depósito a término (CDT). Estos productos son independientes y tienen comportamientos particulares.

El saldo total de la cuenta es la suma de lo que el cliente tiene en cada uno de dichos productos. En la cuenta corriente el cliente puede depositar o retirar dinero. Su principal característica es que no recibe ningún interés por el dinero que se encuentre allí depositado. En la cuenta de ahorro, se paga un interés mensual del 0,6% sobre el saldo. Cuando el cliente abre un CDT, define la cantidad de dinero que quiere invertir y negocia con el banco el interés mensual que va a recibir. A diferencia de la cuenta corriente o la cuenta de ahorro, en un CDT no se puede consignar ni retirar dinero. La única operación posible es cerrarlo, en cuyo caso, el dinero y sus intereses pasan a la cuenta corriente.



Se quiere que el programa permita a una persona simular el manejo de sus productos bancarios, dándole las facilidades de: (1) hacer las operaciones necesarias sobre los productos que conforman la cuenta, y (2) avanzar mes por mes en el tiempo, para que el cliente pueda ver el resultado de sus movimientos bancarios y el rendimiento de sus inversiones.



- Con el botón marcado como ">>" el usuario puede avanzar mes a mes en el simulador y ver los resultados de sus inversiones.
- Con los seis botones de la parte inferior de la [ventana](#), el usuario puede simular el manejo que va a hacer de los productos que forman parte de su cuenta bancaria.
- En la parte media de la [ventana](#), aparecen el saldo que tiene en cada producto y el interés que está ganando en cada caso.

### 4.3. Caso de Estudio N° 3: Un Triángulo

En este caso se quiere construir un programa que permita manejar un triángulo. Esta figura geométrica está definida por tres puntos, cada uno de los cuales tiene dos coordenadas X, Y. Un triángulo tiene además un color para las líneas y un color de relleno. Un color por su parte, está definido por tres valores numéricos entre 0 y 255 (estándar RGB por Red-Green-Blue). El primer valor numérico define la intensidad en rojo, el segundo en verde y el tercero en azul. Más información sobre esta manera de representar los colores la puede encontrar por Internet. ¿Cuál es el código RGB del color negro? ¿Y del color blanco?

El programa debe permitir: (1) Dibujar el triángulo en la pantalla, (2) calcular el perímetro del triángulo, (3) calcular el área del triángulo y (4) calcular la altura del triángulo.



- Con los tres botones de la izquierda, el usuario puede cambiar los puntos que definen el triángulo, el color de las líneas y el color del fondo.
- En la zona marcada como "Información", el usuario puede ver el perímetro, el área y la altura del triángulo (en píxeles).
- En la parte derecha aparece dibujado el triángulo descrito por sus tres puntos.

## 5. Comprensión y Especificación del Problema

Ya teniendo claras las definiciones de problema y sus distintos componentes, en esta sección vamos a trabajar en la parte metodológica de la etapa de [análisis](#). En particular, queremos responder las siguientes preguntas: ¿cómo especificar un [requerimiento funcional](#)?, ¿cómo saber si algo es un [requerimiento funcional](#)?, ¿cómo describir el mundo del problema? Dado que el énfasis de este libro no está en los requerimientos no funcionales, sólo mencionaremos algunos ejemplos sencillos al final de la sección.

Es imposible resolver un problema que no se entiende.

La frase anterior resume la importancia de la etapa de [análisis](#) dentro del proceso de solución de problemas. Si no entendemos bien el problema que queremos resolver, el riesgo de perder nuestro tiempo es muy alto.

A continuación, vamos a dedicar una sección a cada uno de los elementos en los cuales queremos descomponer los problemas, y a utilizar los casos de estudio para dar ejemplos y generar en el lector la habilidad necesaria para manejar los conceptos que hemos ido introduciendo. No más teoría por ahora y manos a la obra.

### 5.1. Requerimientos Funcionales

Un [requerimiento funcional](#) es una operación que el programa que se va a construir debe proveer al usuario, y que está directamente relacionada con el problema que se quiere resolver. Un [requerimiento funcional](#) se describe a través de cuatro elementos:

- Un identificador y un nombre.
- Un resumen de la operación.
- Las entradas (datos) que debe dar el usuario para que el programa pueda realizar la operación.
- El resultado esperado de la operación. Hay tres tipos posibles de resultado en un [requerimiento funcional](#): (1) una modificación de un valor en el mundo del problema, (2) el cálculo de un valor, o (3) una mezcla de los dos anteriores.

#### Ejemplo 2

**Objetivo:** Ilustrar la manera de documentar los requerimientos funcionales de un problema.

En este ejemplo se documenta uno de los requerimientos funcionales del caso de estudio del empleado. Para esto se describen los cuatro elementos que lo componen.

Nombre	<b>R1: actualizar el salario básico del empleado</b>	<b>Es conveniente asociar un identificador con cada requerimiento, para poder hacer fácilmente referencia a él. En este caso el identificador es R1. Es aconsejable que el nombre de los requerimientos corresponda a un verbo en infinitivo, para dar una idea clara de la acción asociada con la operación. En este ejemplo el verbo asociado con el requerimiento es "actualizar".</b>
Resumen	Permite modificar el salario básico del empleado	El resumen es una frase corta que explica sin mayores detalles el <a href="#">requerimiento funcional</a> .
Entradas	Nuevo salario	Las entradas corresponden a los valores que debe suministrar el usuario al programa para poder resolver el requerimiento. En el requerimiento del ejemplo, si el usuario no da como entrada el nuevo salario que quiere asignar al empleado, el programa, no podrá hacer el cambio. Un requerimiento puede tener cero o muchas entradas. Cada entrada debe tener un nombre que indique claramente su contenido. No es buena idea utilizar frases largas para definir una entrada.
Resultado	El salario del empleado ha sido actualizado con el nuevo salario	El resultado del <a href="#">requerimiento funcional</a> de este ejemplo es una modificación de un valor en el mundo del problema: el salario del empleado cambió. Un ejemplo de un requerimiento que calcula un valor podría ser aquél que informa la edad del empleado. Fíjese que el hecho de calcular esta información no implica la modificación de ningún valor del mundo del problema. Un ejemplo de un requerimiento que modifica y calcula a la vez, podría ser aquél que modifica el salario del empleado y calcula la nueva retención en la fuente.

En la etapa de [análisis](#), el cliente debe ayudarle al programador a concretar esta información. La [responsabilidad](#) del programador es garantizar que la información esté completa y que sea clara. Cualquier persona que lea la [especificación](#) del requerimiento debe entender lo mismo.

Para determinar si algo es o no un [requerimiento funcional](#), es conveniente hacerse tres preguntas:

- ¿Poder realizar esta operación es una de las razones por las cuales el cliente necesita construir un programa? Esto descarta todas las opciones que están relacionadas con el manejo de la interfaz ("poder cambiar el tamaño de la [ventana](#)", por ejemplo) y todos los requerimientos no funcionales, que no corresponden a operaciones sino a

restricciones.

- ¿La operación no es ambigua? La idea es descartar que haya más de una interpretación posible de la operación.
- ¿La operación tiene un comienzo y un fin? Hay que descartar las operaciones que implican una **responsabilidad** continua (por ejemplo, "mantener actualizada la información del empleado") y tratar de buscar operaciones puntuales que correspondan a acciones que puedan ser hechas por el usuario.

Un **requerimiento funcional** se puede ver como un servicio que el programa le ofrece al usuario para resolver una parte del problema.

## Ejemplo 3

**Objetivo:** Ilustrar la manera de documentar los requerimientos funcionales de un problema.

A continuación se presenta otro **requerimiento funcional** del caso de estudio del empleado, para el cual se especifican los cuatro elementos que lo componen.

Nombre	<b>R2: ingresar la información del empleado</b>	<b>Asociamos el identificador R2 con el requerimiento. En la mayoría de los casos el identificador del requerimiento se asigna siguiendo alguna convención definida por la empresa de desarrollo. Utilizamos el verbo "ingresar" para describir la operación que se quiere hacer.</b>
Resumen	Permite al usuario ingresar la información del empleado: datos personales y datos de vinculación a la empresa.	Describimos la operación, dando una idea global del tipo de información que se debe ingresar y del resultado obtenido.
Entradas	- Nombre del empleado. - Apellido del empleado. - Sexo del empleado. - Fecha de nacimiento. - Fecha de ingreso a la compañía Salario básico.	En este caso se necesitan seis entradas para poder realizar el requerimiento. Esta información la debe proveer el usuario al programa. Note que no se define la manera en que dicha información será ingresada por el usuario, puesto que eso va a depender del <b>diseño</b> que se haga de la interfaz, y será una decisión que se tomará más tarde en el proceso de desarrollo. Fíjese que tampoco se habla del formato en el que va a entrar la información. Por ahora sólo se necesita entender, de manera global, lo que el cliente quiere que el programa sea capaz de hacer.
Resultado	Nuevo salario	La operación corresponde de nuevo a una modificación de algún valor del mundo, puesto que con la información obtenida como entrada se quieren modificar los datos del empleado.

## Tarea 2

**Objetivo:** Crear habilidad en la identificación y **especificación** de requerimientos funcionales. Para el **caso de estudio 2**, un simulador bancario, identifique y especifique tres requerimientos funcionales.

## Requerimiento Funcional 1

<b>Nombre</b>	
Resumen	
Entradas	
Resultado	

## Requerimiento Funcional 2

<b>Nombre</b>	
Resumen	
Entradas	
Resultado	

### Requerimiento Funcional 3



Nombre	
Resumen	
Entradas	
Resultado	

## Tarea 3

**Objetivo:** Crear habilidad en la identificación y [especificación](#) de requerimientos funcionales.

Para el [caso de estudio 3](#), un programa para manejar un triángulo, identifique y especifique tres requerimientos funcionales.

## Requerimiento Funcional 1

<b>Nombre</b>	
Resumen	
Entradas	
Resultado	

## Requerimiento Funcional 2

<b>Nombre</b>	
Resumen	
Entradas	
Resultado	

### Requerimiento Funcional 3