1. Caso práctico.

Unidad Didáctica I

Caso práctico

Nos ponemos en situacion. En algún lugar de Andalucía tres jóvenes emprendedores; **Jesús, María y José**, Técnicos Superiores en Desarrollo de Aplicaciones Informáticas, y socios de la empresa **Soluciones Informáticas Andalucía S.C.A.**, se encuentran ante el reto de iniciar una gran tarea.

Se trata de realizar una nueva aplicación informática, para la gestión integral de una de las mayores empresas de su zona, concretamente **Servicios Locales S.L.,** que necesita reformar su sistema informático para adaptarse a las nuevas tendencias y ofrecer más y mejores servicios a sus clientes. La competencia de estos jóvenes informáticos, viene de la mano de una empresa consolidada en la zona y que tiene grandes influencias, **KiloSoftware S.A.**

Soluciones Informáticas Andalucía S.C.A. (en adelante **SI Andalucía**) está irrumpiendo con fuerza en el mercado local de servicios a empresas. El primer trabajo (diseño, confección y mantenimiento de una página WEB) que desarrollaron para una de las mayores empresas del municipio, **Servicios Locales S.L.,** ha sido una importante tarjeta de presentación y actualmente están desarrollando tres páginas más para empresas de la zona, lo que les está permitiendo salir adelante, lentamente pero con optimismo.

Pasos para la resolución de un problema

1.1. Creación de una pequeña empresa

Unidad Didáctica I

Creación de una pequeña empresa

La empresa comienza a funcionar y necesitan ayuda. Para ello deciden contratar a **Victor**, un amigo de **Jesús** al que le gusta la informática y todo lo ha aprendido por sí mismo. Es un fenómeno capaz de localizar cualquier cosa en Internet; tiene una habilidad innata para aprender a utilizar casi cualquier programa en unas pocas horas.

Cuando finalizó la secundaria decidió que no iba a estudiar más, que no iba con él, pero una vez que ha conocido los Ciclos Formativos de Informática se está planteando hacer el Ciclo Formativo de Grado Medio de Explotación de Sistemas Informáticos, sobre todo cuando observa cómo se desenvuelven sus compañeros, los conocimientos que tienen y la seguridad con la que actúan. Ha pensado que este empleo le puede acercar a lo que más le gusta, la Informática.

Pasos para la resolución de un problema

1.2. Nuevos compañeros

Unidad Didáctica I

Nuevos compañeros

Por otro lado, algunos de sus antiguos profesores del IES Aguadulce (de Roquetas de Mar en Almería), le han pedido a **María** que la empresa colabore en la FCT (Formación en Centros de Trabajo) de una de sus alumnas, **Carmen**. Evidentemente **María**, recordando sus días en el Instituto y las prácticas en una empresa de desarrollo de aplicaciones, ha aceptado tras consultar a sus compañeros.

Además **Carmen** puede aportar cosas interesantes, ya que tiene un magnífico expediente académico, es una estupenda persona y la empresa necesitará contratar trabajadores cualificados. Así que si demuestra la suficiente capacidad, seguramente le harán una buena oferta al concluir la fase de prácticas. Todos ellos tienen un objetivo común, y es que quieren dedicarse a la Informática

Pasos para la resolución de un problema

1.3. El proyecto

Unidad Didáctica I

El proyecto

Hace unas semanas entregaron a **Servicios Locales S. L.** un proyecto que les habían pedido para la elaboración de una aplicación de gestión de personal y confección de las nóminas de todos sus trabajadores. Este proyecto, ajustado a los requisitos que pedía la empresa, ha superado al resto de los presentados en dos cosas principalmente:

- Tiempo de entrega. Que ellos han establecido en dos meses, adelantando su entrega en un mes respecto a la siguiente mejor oferta.
- Formación para los trabajadores de la empresa. Que incluyen totalmente gratuita durante los dos primeros meses desde la puesta en funcionamiento de la aplicación.

Nuestros amigos están convencidos de que pueden desarrollar una aplicación que cumpla totalmente las expectativas del cliente, con facilidad de uso, fiabilidad y con una documentación y estructura que permitan una inmediata adaptación a futuros cambios en la empresa o en la legislación. Además han añadido algunas mejoras que permitan su inmediata integración con futuras aplicaciones de contabilidad o gestión general de la empresa.

Algunos días después fueron citados **Jesús**, **José** y **María** para entrevistarse con los representantes de **Servicios Locales S.L.** para la aprobación del proyecto correspondiente a la elaboración de una aplicación totalmente personalizada para la gestión de personal de la empresa **Servicios Locales S.L.** Éstas son buenas noticias y lo más inmediato es ponerse a planificar el trabajo y formar un equipo que en principio se centre en este proyecto. Deciden que **José** será la persona que dirija el proyecto y que formarán equipo con él **Víctor** y **María.**

Nuestros protagonistas tienen ante sí un problema por resolver, de una envergadura considerable y deberán abordarlo de una forma sistemática para conseguir los resultados esperados. En este caso se trata de un problema al que se quiere dar una solución informática, y éste es justamente el tipo de problemas que se encargan de resolver los técnicos superiores en Desarrollo de Aplicaciones. Por ello deberán seguir una serie de pasos que en gran medida son más o menos aplicables a la resolución de la mayoría de los problemas que se nos presentan en nuestra vida, pero que tienen sus peculiaridades para el caso de que queramos que nuestra solución pueda ejecutarse en un ordenador.

Una vez en la empresa, se reúne por primera vez el equipo de trabajo del proyecto; Gestión de Personal de Soluciones Locales (GPSL) formado por **José**, **Carmen** y **Víctor**, junto a **María** para un apoyo inicial. **José** va a ser el director del proyecto y como tal, comienza explicando el método de trabajo. Comenta que en principio él irá elaborando los algoritmos y una documentación precisa de todo lo que deben desarrollar basándose en el proyecto de Análisis Previo que ha aprobado la empresa.

José les comenta que de un acertado desarrollo de esta parte va a depender la calidad del software elaborado. **Carmen** sabe de qué está hablando **José** ya que lo ha escuchado continuamente a sus profesores en clase, pero **Víctor** no está muy de acuerdo, él prefiere sentarse ya ante el ordenador y comenzar a programar. **José** les explica que el plan de desarrollo va a consistir en que una vez que él elabore los algoritmos, **Víctor** y **Carmen** comenzarán con la codificación en Java de cada algoritmo, seguirán con la compilación y pasarán a la fase de prueba para comprobar que los programas desarrollados cumplen con las especificaciones requeridas.

Pasos para la resolución de un problema

2. Pasos para la resolución de un problema.

Unidad Didáctica I

Pasos para la resolución de un problema.

Preguntas propias de Victor.
¿Es necesario analizar los problemas para hacer un programa de ordenador?
En primer lugar, hay que decir que la programación de ordenadores es una solución a determinados problemas. Pero para que esta solución sea eficaz es preciso conocer profundamente el problema que pretendemos solucionar y la mejor forma de hacerlo es siguiendo un proceso sistemático de estudio que permita tener en cuenta todos los casos posibles.
¿Cuánto tiempo se pierde en preparar un programa?
El tiempo que se dedica a preparar un programa nunca es tiempo perdido, ya que consiste en conocer todos y cada uno de los casos posibles y ello siempre redunda en la calidad del software (comprensión, facilidad de uso, facilidad de mejora, eficiencia, eficacia y perfecto control de excepciones).
Por ejemplo, imaginemos que se nos ha pedido que diseñemos una aplicación informática para el cálculo del Precio de Venta al Público de diferentes productos en función del tipo de IVA a aplicar.

El motivo por el que montones de personas aprenden lenguajes de programación es para poder usar los ordenadores como una herramienta para resolver los problemas. Consideramos que la solución de un problema comienza con la **definición** del mismo y termina con la verificación de que la solución encontrada es válida para todas las situaciones posibles.

La ***fase de resolución*** abarca hasta ese punto.

Siguiendo nuestro ejemplo esta fase consistiría en entender bien el cálculo, cada uno de los tipos de IVA posibles, las situaciones en las que se aplican y a qué productos. Luego en conocer el método de cálculo (en este caso sumar el porcentaje correspondiente al precio del producto) y elaborar un algoritmo adecuado que permita llegar a una solución correcta. Finalmente una vez elaborado este algoritmo es preciso comprobar que es correcto y que los cálculos son exactos.

Pero en el caso de una solución informática, hay que ir un poco más lejos, ya que deberemos:

- "traducir" esa solución para que sea comprensible y ejecutable por un ordenador,

- corregir los errores que contenga y
- probar que su funcionamiento es el correcto.

Hasta aquí abarca la ***fase de implementación***.

En el problema del cálculo del IVA, esta fase consistiría en codificar el algoritmo en un lenguaje de programación (por ejemplo Java, C, Basic, Pascal, etc.), para después compilarlo y una vez depurados todos los errores hacer todo tipo de pruebas para comprobar que su funcionamiento sea el esperado.

Pero incluso una vez que nuestra aplicación (o programa de ordenador) está terminada hay que procurar mantenerla actualizada, haciéndole ajustes, mejoras, adaptaciones, etc. siempre que sean necesarias. Esa es la ***fase de mantenimiento***.

Esta fase en nuestro ejemplo del IVA, al ser tan sencillo podemos plantearla desde el punto de vista de alguien al que le gustan las aplicaciones con determinados colores, con sistemas de entrada de datos más sofisticados (códigos de barras) a través de bases de datos, o incluso la adaptación de este pequeño programa para su uso en aplicaciones mayores.

Naturalmente cada una de las fases enunciadas conviene dividirla a su vez en una serie de pasos que faciliten su desarrollo de forma más sencilla, cómoda y eficaz.

Para saber más:
En el siguiente enlace, aparece un artículo en el que se ejemplifica de forma más o menos general, los pasos que deben darse para solucionar un problema en general. La mayoría de ellos son los mismos que aplicamos cuando intentamos solucionar un problema informático.
Pasos para solucionar un problema. **[Versión en Caché]**

(Si tienes problemas para acceder a algún enlace, pulsa en "Versión en Caché" para visualizar una copia de esa página web)
Para descargar el programa Acrobat Reader pulsa aquí.

AUTOEVALUACIÓN:

La división en pasos de la resolución de un problema pretende:

a) Dividir el trabajo en distintas partes, de forma que cada una de ellas pueda asignarse por separado a un grupo distinto de personas, especializado en esa tarea.
b) Establecer una forma sistemática de abordar la solución de un problema, para que aunque éste sea de envergadura, se facilite el trabajo de encontrar la solución, y se garantice que el trabajo se hace correctamente.
c) Las respuestas a y b son ciertas.
d) La respuesta b es cierta, pero no lo es a.

Comprobar

Pasos para la resolución de un problema

3. Fase de Resolución.

Unidad Didáctica I

Fase de Resolución

***Carmen**, estudiante de Ciclo Formativo de Desarrollo de Aplicaciones Informáticas en prácticas, recuerda las palabras de su profesor de programación cuando le decía que la parte más importante de un programa es la fase de resolución, que es la que condiciona todo el trabajo posterior y va a determinar la manera de trabajar de un equipo. Ahora, cuando se encuentra ante el análisis de una aplicación reconoce la dificultad de esta parte y entiende que para trabajar en equipo es necesario que todos sigan las mismas pautas.*

En esta fase **tratamos de acercarnos al problema**, **definirlo correctamente, e idear una forma de resolverlo**. También deberemos verificar que la solución ideada es válida para todos los casos posibles. Todo ello con independencia de que esa solución se vaya a llevar a cabo manualmente, con papel y lápiz, o con un ordenador. Intentamos describir esa solución de forma genérica, sin que todavía entremos en detalles asociados a un lenguaje de programación concreto o a las características de un ordenador concreto. **El resultado de esta fase será un algoritmo, ya verificado**, expresado mediante alguna herramienta descriptiva. Los pasos que incluye esta fase (y que se muestran en la imagen) se detallan en los apartados siguientes.

Pasos para la resolución de un problema

3.1. Análisis del problema.

Unidad Didáctica I

Análisis del problema.

__José__ intenta convencer a __Víctor__ de que el análisis del problema no es una pérdida de tiempo y harto de dar explicaciones decide demostrárselo dándole una lección. Para ello le reta a que demuestre que su método de "programación directa" es mejor.
El resultado es obvio. __Víctor__ consigue hacer una aplicación que va retocando para tener en cuenta todas las excepciones del problema. Finalmente __José__ consigue terminar su aplicación mientras su amigo sigue haciendo retoques. Las pruebas demuestran que el programa elaborado por __José__ funciona perfectamente, mientras que el de __Víctor__ está incompleto y presenta algunos problemas de incoherencias en determinados resultados que debe revisar.
En este momento __Jesús__ plantea dos nuevas excepciones que debe tener en cuenta el programa y que requieren algunas modificaciones en la aplicación ya desarrollada. La adaptación de la aplicación de __José__ es inmediata, añade comentarios a la documentación, hace algunos ajustes en el análisis y finalmente modifica el código. Las pruebas finales demuestran que esta aplicación cumple con los requerimientos del cliente y además está perfectamente preparado para futuras actualizaciones, mejoras o cambios.
Parece que finalmente __Víctor__ se ha convencido de que los métodos de programación recomendados por los expertos permiten obtener resultados óptimos. Y se ve obligado a dar la razón a __José__ de que la programación de ordenadores debe comenzar con lápiz, papel y goma, por muy avanzadas que se encuentren las nuevas tecnologías.

Aquí daremos una descripción muy superficial de esta fase, ya que es objeto de estudio del módulo profesional de Análisis y Diseño Detallado de Aplicaciones Informáticas de Gestión, dentro del ciclo formativo de DAI, tan extenso por cierto como el de Programación en Lenguajes Estructurados, que ahora es el que nos ocupa.

El análisis es la primera aproximación al problema. Consiste en **estudiar el problema**, asegurándonos de que lo entendemos bien, de que hemos tenido en cuenta todos los casos y situaciones posibles, etc. Una vez comprendido el problema, será necesario dar una definición lo más exacta posible del mismo, identificando qué tipo de **información** se debe producir y qué **datos** o elementos dados en el problema pueden usarse para obtener la solución.

Como resultado de un buen análisis del problema, obtendremos un **conjunto de especificaciones de entrada y de salida**, que nos definen los **requisitos que nuestra solución** al problema debe cumplir.

Las especificaciones de entrada deben responder a preguntas del tipo:

- ¿Qué **datos** son de entrada?
- ¿Cuál será el volumen de **datos** de entrada?
- ¿Cuándo se considerará que un dato de entrada no es válido?
- ¿En qué soporte están los **datos** de entrada?

Las especificaciones de salida deben responder a preguntas del tipo:

- ¿Cuáles son los **datos** de salida?
- ¿Cómo se obtienen los **datos** de salida a partir de los de entrada?
- ¿Qué volumen de **datos** de salida se producirán?
- ¿Qué precisión deberán tener los resultados?
- ¿Cómo deben presentarse los **datos** de salida?
- ¿En qué soporte se almacenarán esos **datos**, si es que se almacenan?

Con esto ya debemos tener bastante claro nuestro problema, que está **definido de forma precisa y**

sin ambigüedades, e incluso debemos tener alguna idea de cómo resolverlo.

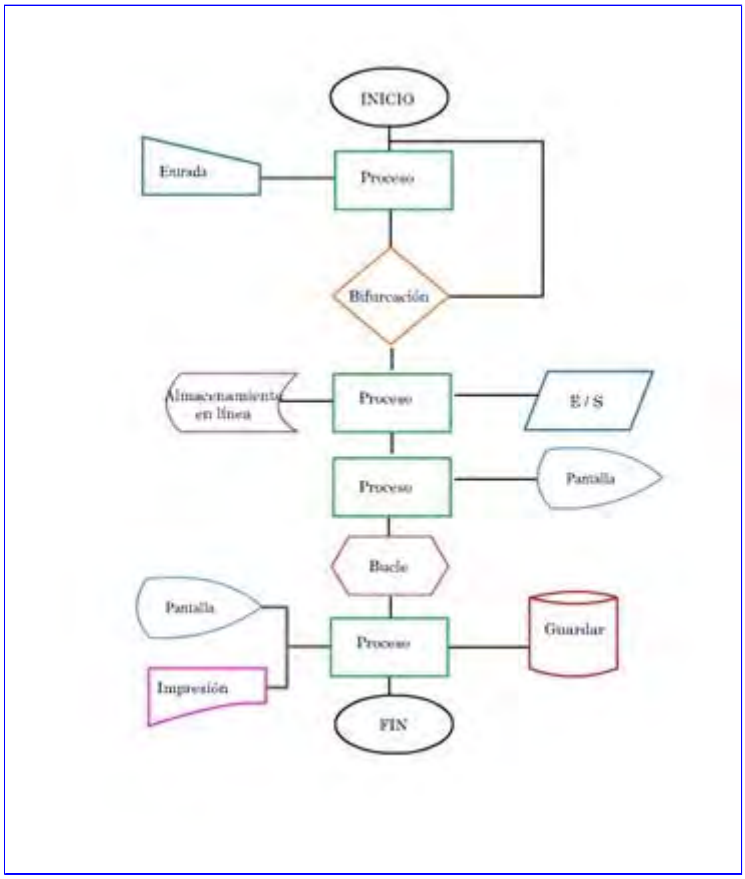
Pasos para la resolución de un problema

3.2. Diseño del Algoritmo.

Unidad Didáctica I

Diseño del Algoritmo.

__José__ explica a su amigo __Víctor__ que esto es un equipo y como tal, todos deben seguir los mismos procedimientos, emplear una misma terminología para referirse a los diferentes elementos y hacer una misma interpretación de cada una de las fases del proceso. Por ejemplo __José__ elaborará los algoritmos de cada uno de los programas de la aplicación y los programadores deben interpretar correctamente las indicaciones de los mismos para obtener el código esperado. Por ello es necesario que __Víctor__ conozca la confección de algoritmos y no estaría de más, para empezar, que intentara desarrollar alguno no muy complejo.



En este paso se trata de idear y representar de forma más o menos normalizada**, la solución del problema**. Para ello se usan métodos descriptivos o herramientas estándar, que puedan ser entendidos por cualquier **programador**, y que no admitan ambigüedades ni distintas interpretaciones, pero sin entrar en detalles de sintaxis del **lenguaje de programación**.

La idea es describir la solución al problema (algoritmo) con la claridad suficiente como para que trasladar la solución a cualquier lenguaje de programación concreto sea inmediata.

En esta fase, frecuentemente nos encontraremos con la necesidad de resolver un problema complejo, en cuyo caso lo más adecuado es usar la técnica de "**divide y vencerás**", que consiste en **dividir el problema en subproblemas más sencillos, que pueden ser a su vez subdivididos hasta conseguir problemas de fácil solución.**

Esta técnica se conoce también como ***diseño descendente***, o ***diseño por refinamiento paso a paso o sucesivo***, ya que partimos de una descripción de un problema global, que en un primer refinamiento lo describimos como formado por varios subproblemas, cada uno de los cuales se irá detallando en cada refinamiento sucesivo, hasta llegar a los subproblemas básicos, que no necesitan ser subdivididos.

Se llama diseño descendente porque partimos de lo más complejo hasta llegar a lo más simple, descendiendo en nivel de dificultad en cada paso o refinamiento.

Para saber más:
Lee el texto del siguiente enlace, correspondiente a un curso de la Universidad de Manizales, de Colombia. En él se ofrece una reflexión sobre algunos de los aspectos que se mencionan en la definición de algoritmo.

Algoritmo **[Versión en Caché]**

(Si tienes problemas para acceder a algún enlace, pulsa en "Versión en Caché" para visualizar una copia de esa página web)

El estudio de estas herramientas y metodologías de diseño será abordado en la unidad didáctica 4, y con más detalle en el módulo profesional de Análisis y Diseño Detallado de Aplicaciones Informáticas de Gestión, dentro del ciclo formativo de DAI.

Pasos para la resolución de un problema

3.3. Verificación "manual" del algoritmo.

Unidad Didáctica I

Verificación "manual" del algoritmo.

Víctor se ha animado a elaborar un algoritmo pero evidentemente tiene sus dudas de que realmente funcione y se ajuste a lo que se necesita. José le explica que una vez concluido el algoritmo es necesario hacer una verificación, algo parecido a la prueba de una división, ya que si comenzara a codificarlo sin haberlo verificado y luego es incorrecto, esa codificación no serviría de nada y habría perdido el tiempo. Le explica que debe entender que estos métodos han sido contrastados por grandes programadores y han superado las pruebas más estrictas de personas muy inteligentes, que han sido determinantes en lo que es ahora la Informática. Parece que poco a poco Víctor se va convenciendo, y llega a la conclusión de que estudiar cualquier materia te abre nuevos horizontes, que te hace más humilde y te enseña a respetar las opiniones de otras personas.

Una vez que hemos descrito el algoritmo que da solución a nuestro problema, mediante alguna herramienta descriptiva (o lenguaje algorítmico) es necesario asegurarse de que efectivamente realiza todas las tareas para las que se ha diseñado de forma correcta, produciendo el resultado esperado. Para ello será necesario "ejecutar" manualmente el algoritmo, **probándolo para un conjunto de datos que incluyan todos los casos posibles,** incluidos los más extremos y los menos frecuentes, abarcando todo el rango de valores de entrada posibles. Deberemos ir anotando en una hoja los valores intermedios de todos los **datos** que se vayan calculando, y los resultados finalmente obtenidos.

Es muy posible que durante la verificación del algoritmo nos demos cuenta de que algunos fallos se deben a un mal diseño del algoritmo, o incluso a un análisis equivocado. Esto nos puede llevar a **volver a cualquiera de estas fases anteriores, bien añadiendo o modificando especificaciones, o bien proponiendo cambios en la manera de resolver el problema**. Después, sea cual sea el caso, deberemos volver a verificar el algoritmo, como es lógico, hasta que supere todas las pruebas realizadas.

En la unidad didáctica 4, se explica más a fondo el concepto de algoritmo, y se ponen ejemplos tanto del análisis, como del diseño y de la verificación de algunos algoritmos. Tú también tendrás que realizar algunos como actividades, pero por ahora solo estamos contando los pasos que hay que dar. Aún no hemos empezado a andar. De momento es conveniente que sepas de qué hablamos, así que te proponemos que pruebes la siguiente aplicación en la que se explica el funcionamiento de un algoritmo para el cálculo del Precio de Venta al Público, dado su precio y el tipo de IVA que se le aplica. Se trata de una simulación de la verificación manual del algoritmo, tal y como debe llevarla a cabo el analista aplicándola a dos ejemplos en los que se detalla el comportamiento del algoritmo ante diferentes valores de entrada

Aplicación demostrativa de Verificación manual de Algoritmo. Pulse sobre la imagen.

AUTOEVALUACIÓN:

El análisis del problema pretende:

a) Mejorar el conocimiento del problema, estudiarlo, dar una definición formal del mismo, y establecer la información de entrada y de salida que debe manejar el programa.

b) Comprobar que la solución del problema no tiene errores.

c) Evaluar la eficiencia del algoritmo empleado en la solución.

d) Codificar el programa a partir de las especificaciones de entrada y de salida.

Comprobar

La tarea de comprobar que el algoritmo ideado resuelve correctamente el problema planteado, produciendo el resultado esperado, y para todos los casos posibles recibe el nombre de:

a) Verificación manual del algoritmo.

b) Implementación del algoritmo.

c) Codificación del algoritmo.

d) Depuración del algoritmo.

Comprobar

Pasos para la resolución de un problema

4. Fase de Implementación.

Unidad Didáctica I

Fase de Implementación.

*De la implementación se van a encargar **Carmen** y **Víctor** una vez que **José** ha concluido los algoritmos y les ha dado una serie de directrices sobre cómo abordarlos. Para ello, deben ser capaces de convertir el algoritmo en un programa con código en Java libre de errores. **Carmen**, aunque nunca ha participado en un grupo de trabajo como este, demuestra cierta soltura en la codificación de los algoritmos y en el dominio del lenguaje de programación elegido, Java. Por otro lado, **Víctor** tiene algunas dificultades en los primeros algoritmos, pero cuando **José** le aclara las dudas, parece que no se defiende mal. El resultado obtenido son pequeños programas que hacen tareas específicas y que deben ser ensamblados en un programa mayor. De esto ya se encargará **José**.*

A esta fase llegamos con una solución a nuestro problema, que ya sabemos que es correcta. Y además la tenemos descrita de una forma clara, sin ambigüedades, por lo que si queremos resolver el problema con un ordenador bastará con **pasar (traducir o codificar) del lenguaje algorítmico a un lenguaje de programación concreto**, como por ejemplo Java, o C, o Delphi, o Visual Basic, etc.

En esta fase de implementación, además deberemos corregir los errores del programa escrito en el lenguaje de programación elegido hasta que pueda ejecutarse.

*(**Víctor** y **Carmen** lo escriben en Java y lo compilan comprobando que no han cometido errores)*

Y una vez ejecutado, deberemos hacer pruebas, al igual que con el algoritmo para verificar que todo funciona como se esperaba

En nuestro caso, el programa ejecutable debe ser revisado minuciosamente para que un trabajador no cobre menos de lo que le corresponde, ni la empresa pague más de lo que

debe.

Los pasos a seguir en la fase de implementación son los que se describen en los apartados siguientes.

Pasos para la resolución de un problema

4.1. Codificación.

Unidad Didáctica I

Codificación.

Víctor está entusiasmado con la codificación y comenta con sus compañeros que es como traducir el algoritmo para que lo entienda el ordenador. Añade que al codificar él se imagina cómo interpretará el ordenador lo que está escribiendo, y que eso le divierte mucho. No obstante reconoce que para hacer una buena codificación es imprescindible dominar el lenguaje, y ése precisamente es su punto débil, pero como es un chaval rebosante de optimismo, espera conseguirlo muy pronto, aunque tenga que estudiar para ello.

Este paso consiste en pasar de la descripción del algoritmo a un lenguaje de programación concreto, tal como Java. Una vez que hayamos elegido el lenguaje de programación concreto, sólo tendremos que ir siguiendo el algoritmo, y **escribiendo con la sintaxis de ese lenguaje las sentencias o instrucciones que hacen lo que se indica en el algoritmo**.

Por ejemplo, si el algoritmo indica:

```
nombre ← "Juan"
Si (nombre = "Pepe")
        Escribir ( "El nombre del empleado es ", nombre)
Caso Contrario
        Escribir ("No lo conozco, pero se llama " , nombre)
Fin-Si
```

En Java deberemos escribir algo como lo que sigue:

```
String nombre = "Juan" ;
if (nombre.equals("Pepe")){
        System.out.println("El nombre del empleado es "+ nombre) ;
}else {
        System.out.println("No lo conozco, pero se llama " + nombre) ;
}
```

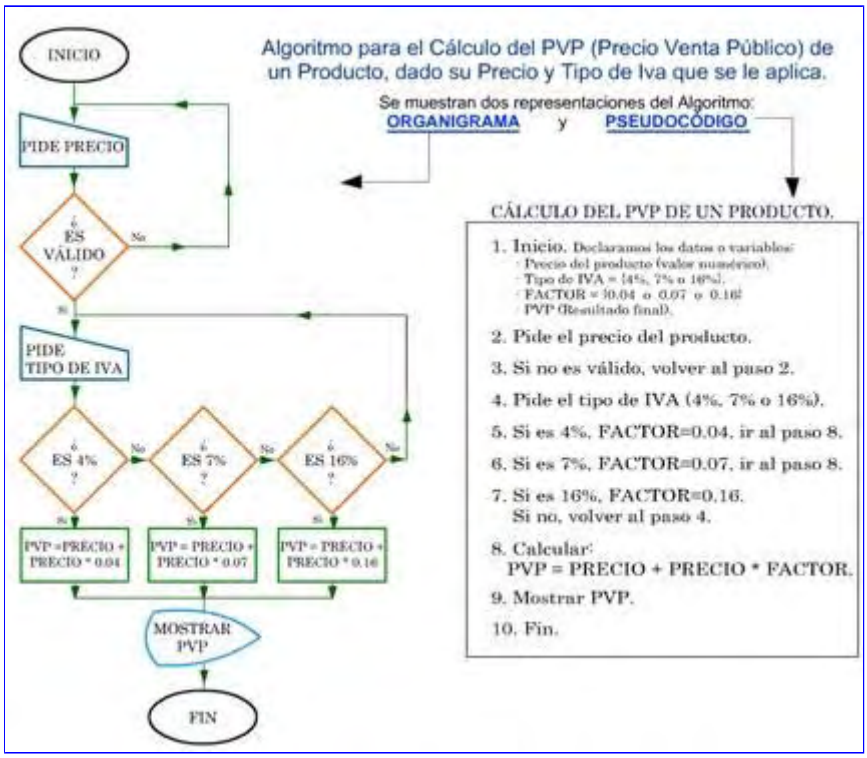
Aún sin conocer todavía la sintaxis de Java ni la forma de describir el algoritmo en lenguaje algorítmico, vemos que existe bastante similitud entre la descripción más o menos formal, pero cercana a nuestra forma de hablar del algoritmo y el código escrito en lenguaje Java. La diferencia es que en el segundo tenemos que tener en cuenta una serie de detalles de **sintaxis del lenguaje**, tales como que las sentencias terminan con punto y coma, o las llaves para delimitar bloques de sentencias, o las palabras concretas que usamos para comprobar si se cumple o no una condición, o la forma de asignar un valor a una variable... En Java la sintaxis es esa y hay que escribirlo así, porque de otro modo, no funcionaría, el ordenador no lo entendería.

Por el contrario, en el algoritmo la sintaxis no es nada rígida. Podíamos haberlo expresado de otra forma, ya que el único requisito a cumplir es que quede claro para cualquier persona lo que hay que hacer en cada paso y el orden en el que se deben dar, sin ambigüedades.

Dicho de otra forma, **el lenguaje algorítmico es independiente del lenguaje de programación**, y por tanto no tiene porqué atenerse a una sintaxis determinada, por lo que cada persona puede expresarlo de forma distinta, siempre y cuando cumpla con las **dos reglas**:

- Sea fácil de comprender por cualquier persona.
- No sea ambiguo.

A continuación mostramos dos formas de representar el algoritmo para el cálculo del PVP de un producto. Se trata del mismo que se ha utilizado en la demostración del apartado de verificación manual del algoritmo. Estas dos representaciones del algoritmo nos pueden llevar al mismo código en el lenguaje de programación elegido. Por tanto la codificación, por ejemplo en Java, va a consistir en la traducción de cada uno de los pasos de una de estas representaciones de la solución, a instrucciones del lenguaje, obteniendo como resultado un programa que posteriormente se escribirá en el ordenador para su compilación y ejecución.



Dos formas de representar una solución al problema.

Pasos para la resolución de un problema

4.2. Traducción a código máquina.

Unidad Didáctica I

Traducción a código máquina.

*Cuando **Carmen** y **Víctor** concluyen la codificación de cada uno de los programas, es necesario pasar por un proceso automático que permita que el ordenador entienda el código. Esto se hace a través de los programas compiladores e intérpretes que tomando un código escrito en un lenguaje de programación concreto lo traducen a código máquina (ceros y unos). Pero como dice **Víctor**; "eso a nosotros no nos importa, yo me hago a la idea de que el ordenador entiende el código que yo escribo. ¡Y me funciona!". Por una vez y sin que sirva de precedente **José** está de acuerdo con él.*

Todos los lenguajes de programación que se usan normalmente son llamados **lenguajes de alto nivel,** en alusión a su parecido con nuestra forma de pensar y de expresarnos, aunque con una sintaxis mucho más rígida, que no de lugar a posibles interpretaciones, y que defina de forma clara lo que el ordenador debe hacer en cada momento. Incluso para construir las sentencias usan un buen número de palabras de nuestro idioma (aunque ese idioma es normalmente inglés).

Los lenguajes de alto nivel tienen como características básicas que son más fáciles de comprender y su portabilidad, o independencia del tipo de ordenador en el que se vayan a ejecutar los programas.

No obstante, los ordenadores funcionan con corriente, y eso es lo único que pueden entender. Concretamente, funcionan con sólo dos valores de corriente distintos y bien diferenciados (Por Ejemplo: En una línea de un circuito, en un momento dado hay una tensión de +5V o una tensión de -5V, y ningún otro valor es posible). **A esos dos valores distintos los identificamos como 0 y 1**, y decimos que los ordenadores son binarios, o que trabajan en binario. Pues bien, cualquier instrucción que queramos que ejecute el ordenador, deberemos expresársela en forma de ceros y unos, que es lo único que entiende. Ese lenguaje es lo que llamamos **lenguaje máquina**, y decimos que es un **lenguaje de bajo nivel**, ya que está alejado de nuestra forma de pensar y expresarnos.

Pero, ¿quién realiza esa traducción de lenguaje de alto nivel a código máquina? La respuesta es que **la realizan automáticamente unos programas llamados Traductores**. Existen dos tipos de traductores, que son los **compiladores** y los **intérpretes**.

Pasos para la resolución de un problema

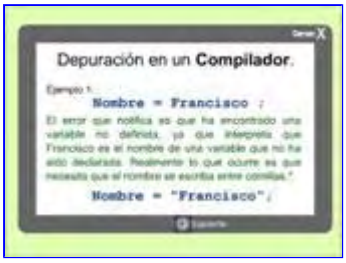
4.3. Compiladores.

Unidad Didáctica I

Compiladores.

Víctor *dice que para él esta fase consiste en preguntarle al ordenador si entiende todo lo que se le está diciendo, a lo que este responde con un "Ok. Lo entiendo todo", o con una lista de errores en los que indica aquellos tokens (símbolos) que no entiende, y por consiguiente no puede seguir con el programa.* ***José*** *le dice que aunque sea un buen símil es conveniente que utilice la terminología adecuada porque de lo contrario no le van a tomar en serio, le corrige para que utilice el término "Depuración de errores", aunque reconoce que no deja de ser ocurrente este chico.*

La analogía puede establecerse con el traductor que traduce un libro de inglés a español, produciendo un ejemplar escrito en español, que puede leerse tantas veces como se quiera sin tener que volver a traducirlo.



Un compilador es un programa traductor al que se le pasa como dato de entrada el fichero de texto con el programa escrito en lenguaje de alto nivel, al que se llama **código fuente** (por ejemplo, fichero con el programa escrito en lenguaje C). El compilador analiza todo el texto, y si encuentra errores, nos da un listado completo de los errores encontrados, para que editemos el código fuente y los corrijamos. Este proceso se repetirá mientras el programa contenga errores, se denomina **depuración de errores**, Si por el contrario no hay errores, **efectúa la traducción de todo el**

programa a código máquina, proporcionando el resultado grabado en un fichero al que se le denomina código objeto, y que es ejecutable tantas veces como se desee, sin tener que volver a traducirlo, por lo que se ejecutará más rápidamente.

En la imagen podemos ver las fases de un compilador. Aunque en ella apreciamos que se trata de un proceso complejo, como programadores no necesitamos entrar en tantos detalles. Nos basta saber que **los compiladores traducen todo el programa escrito en un determinado lenguaje de alto nivel**, obteniéndose como resultado un fichero con el programa traducido a código máquina que es ejecutable sin tener que volver a traducir.

Tienen la desventaja de que en el proceso de depuración de los errores, al intentar analizar y traducir todo el texto y generar un listado con todos los errores encontrados, a veces se incluyen en ese listado errores que no son reales o mensajes de error poco claros, que se deben a que un error previo hace que el compilador no termine de entender correctamente el código que hay detrás de ese error. Pulse sobre la imagen para ejecutar una aplicación con ejemplos de aclaración de errores detectados por el **compilador**.

En el caso particular del lenguaje **Java**, no se trata de un lenguaje compilado, si no pseudocompilado, ya que al compilar ***no*** traduce directamente a código máquina, sino a un código intermedio que denomina **Bytecodes**, y que sigue siendo portable, independiente de la máquina, es decir, independiente del microprocesador concreto que tenga el ordenador en el que se ejecute.

Para saber más:

En los siguientes enlaces encontrarás información algo más detallada acerca de las características de los compiladores y de los lenguajes compilados.

Características de los compiladores **[Versión en Caché]**

Lenguajes Compilados **[Versión en Caché]**

En este enlace aparece información exhaustiva sobre el funcionamiento de los compiladores.

Funcionamiento de Compiladores **[Versión en Caché]**

(Si tienes problemas para acceder a algún enlace, pulsa en "Versión en Caché" para visualizar una copia de esa página web)

Pasos para la resolución de un problema

4.4. Intérpretes.

Unidad Didáctica I

Intérpretes.

Víctor *no termina de entender la diferencia entre compiladores e intérpretes y dice que no es lógico que existan dos términos para definir el mismo concepto.* ***Carmen*** *le explica que aunque en el fondo ambos conceptos "traducen" el código fuente (en un lenguaje determinado) a código máquina (ceros y unos) la diferencia está en la forma de hacerlo. Mientras el compilador revisa todo el código de principio a fin y devuelve todos los errores encontrados en bloque, el intérprete lo hace instrucción a instrucción e interrumpe todo el proceso cuando encuentra un error sin revisar las líneas de código siguientes.*

Victor *insiste que hacen lo mismo y que prefiere el intérprete, porque es más rápido.* ***Carmen*** *le dice que eso es una apreciación equivocada y que depende de la situación uno presentará más ventajas que el otro. Pero que habitualmente los programadores profesionales utilizan compilador.*

Puede establecerse una analogía de los programas Intérpretes con el traductor simultáneo, al que se le da un libro en inglés para que vaya leyendo en español lo que en él dice. Después de haberlo leído, si quiero volver a escuchar lo que dice el libro en español, tengo que volver a pedirle al traductor que venga a leerlo.

Un intérprete es un programa traductor al que se le pasa como dato de entrada el fichero de texto con el programa escrito den lenguaje de alto nivel (**código fuente**) pero en vez de hacerlo de golpe y producir una versión ejecutable en **código máquina**, lo que hace es leer una a una las instrucciones del programa, analizarla, y si contiene errores parar el proceso y notificarlo para editar esa instrucción en el código fuente. Si no hay errores, **la traduce a código máquina en memoria, y la ejecuta**, sin guardar la traducción en ningún sitio. Por tanto, si quiero volver a ejecutar de nuevo el programa, deberé volver a traducir una a una cada línea antes de ejecutarla, por lo que la ejecución será más lenta. Por el contrario, el tiempo invertido en la depuración suele ser menor, ya que los mensajes de error suelen ser más precisos y estar más localizados.

Para saber más:

El siguiente enlace ofrece información adicional sobre las características de los intérpretes y los lenguajes interpretados.

Lenguajes Interpretados **[Versión Caché]**

(Si tienes problemas para acceder a algún enlace, pulsa en "Versión en caché" para visualizar una copia de esa página web)

AUTOEVALUACIÓN:

El proceso de traducción a código máquina:

a) Se realiza como parte de la optimización del código, para aumentar la rapidez en la ejecución del programa.

b) Consiste en el paso del programa de código fuente a un lenguaje binario, compuesto por ceros y unos, con el objetivo de mejorar la portabilidad del código.

c) Resulta imprescindible, ya que el código máquina es el único lenguaje directamente ejecutable por los ordenadores, por lo que cualquier programa escrito en cualquier otro lenguaje, deberá ser traducido a código máquina antes de ser ejecutado

d) Todas las respuestas anteriores son correctas

Comprobar

El tipo de traductores que realizan un análisis sentencia a sentencia del código fuente, traduciendo a código máquina cada una de ellas y ejecutándola inmediatamente antes de analizar y traducir la siguiente, sin guardar en ningún fichero esa traducción es:

- a) Ensambladores
- b) Compiladores
- c) Intérpretes
- d) Código Máquina

Comprobar

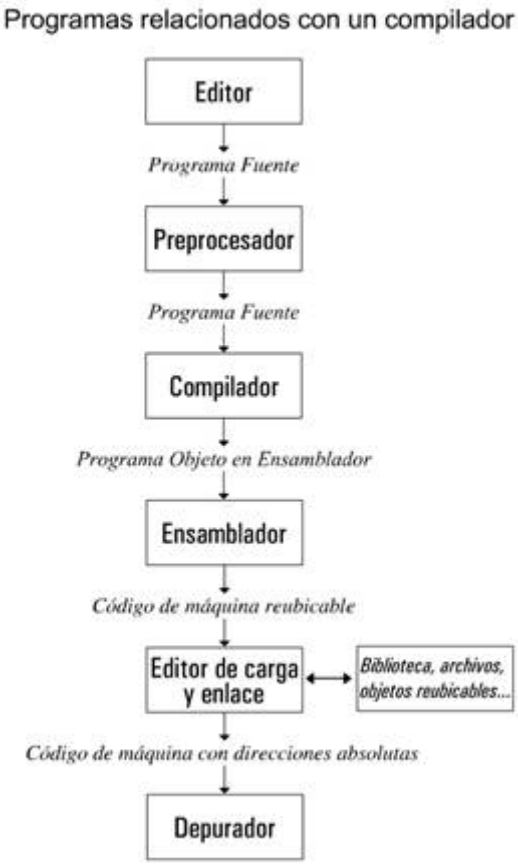
Pasos para la resolución de un problema

4.5. Depuración.

Unidad Didáctica I

Depuración.

*Una vez elaborado el programa **Víctor** se siente satisfecho y no para de comprobar su funcionamiento, quiere contárselo a todo el mundo y llama a **Carmen** para enseñarle su primer programa. **Carmen** le pide que tenga paciencia que está en fase de Depuración, **Víctor** se sorprende, no sabe qué es eso y obligado por su insaciable curiosidad se dirige al puesto de **Carmen** y le pregunta al respecto. Ésta le contesta que está resolviendo algunos errores que ha encontrado al hacer la compilación, y que estos errores una vez resueltos hay que volver a compilar para ver si el ordenador "comprende" todo lo que se le pide.*



Ya tenemos el programa escrito en el lenguaje de programación, pero como somos humanos, nos hemos podido confundir, por ejemplo:

- olvidando un punto y coma al final de una sentencia,
- no cerrando un paréntesis,
- olvidando escribir una sentencia,
- escribiéndola mal, etc.

Todos esos errores deberán ser corregidos para que el programa pueda ser ejecutado por el ordenador. El proceso de depuración incluye la realización de un ciclo de tareas, hasta que el

resultado es correcto. **Ése ciclo consiste fundamentalmente en:**

1. **compilar el programa**
2. **analizar el listado de errores**
3. **editar y corregir el texto del programa**
4. **volver a compilar el programa.**

Para entender ese proceso hay que detenerse un momento a explicar el significado de la palabra **compilar.** Mira la imagen para entender el camino que sigue un programa durante la compilación.

Pasos para la resolución de un problema

4.6. Pruebas.

Unidad Didáctica I

Pruebas

__Víctor__ ayuda a __Carmen__ en la Depuración y una vez libre de errores, da por concluido su trabajo, pero __Carmen__ le indica que esto no ha terminado, que aunque la aplicación funcione es necesario comprobar que responde adecuadamente ante todas las posibles situaciones que se pueden presentar, por lo que es necesario plantear una serie de pruebas completas que verifiquen el correcto funcionamiento de la aplicación. __Carmen__ añade que ese trabajo normalmente lo hace la persona que ha elaborado el algoritmo y que en su caso __José__ tiene las pruebas a las que es necesario someter al programa para tratarlo como APTO. __Víctor__ se siente cada vez más desesperado y pregunta que si es que aquí no se acaba nunca, a lo que responde __María__, que pasaba por allí, que un programa nunca se tiene la certeza de haberlo concluido al 100% y que en cualquier momento pueden aparecer problemas en su funcionamiento. Añade que ese es el motivo de hacer un mantenimiento o seguimiento una vez concluido, pero que eso hay que evitarlo en lo posible sometiendo al programa a todo tipo de inclemencias.

Una vez finalizada la depuración, dispondremos de la posibilidad de ejecutar el programa. Ahora bien, es posible que aunque el programa funcione no haga exactamente lo que se necesita, o que funcione mal para algún dato de entrada, etc.

Será por ello necesario probar el funcionamiento del programa con un conjunto de datos lo suficientemente significativos, incluyendo valores extremos o raros.

También debemos asegurarnos de haber probado toda la casuística prevista en el programa, de forma que no queden bloques de código sin someter a su oportuno test de prueba. **Cualquier fallo detectado nos puede llevar a replantear cualquiera de las etapas anteriores**, incluyendo el análisis o el diseño del algoritmo, o la codificación. No obstante, lo más frecuente es que se trate de un error en la etapa inmediata anterior.

También debemos señalar que los tests de pruebas deben diseñarse como parte de las especificaciones del análisis (en la fase de resolución), y no después de haber desarrollado la aplicación. A ser posible, incluso por otra persona que no haya sido el que ha desarrollado la aplicación. El motivo es que cuando nosotros desarrollamos y codificamos una aplicación en parte estamos viciados incluso inconscientemente por nuestro conocimiento de su funcionamiento interno, y tendemos de forma involuntaria a probarla sólo con los valores para los cuales la hemos pensado y para los casos en los que sabemos que funciona. Si el conjunto de pruebas se elaboró previamente, y por otra persona, tal circunstancia no es posible.

AUTOEVALUACIÓN:

En la realización de las pruebas, durante la fase de Implementación:

a) Cualquier error encontrado nos llevará de nuevo a revisar la codificación del algoritmo.

b) Cualquier error encontrado nos llevará a revisar el diseño del algoritmo.

c) Cualquier error encontrado nos llevará a revisar el análisis del

problema.

- d) Cualquier error encontrado nos llevará, dependiendo de la gravedad del mismo, a revisar cualquiera de las etapas anteriores, incluyendo el análisis del problema, el diseño del algoritmo, la verificación del algoritmo, la codificación, la depuración, la propia fase de pruebas de la aplicación o incluso todas ellas

Comprobar

Los tests de pruebas deberán ser diseñados:

- a) Por el equipo de programadores, que son los que mayor conocimiento tienen del funcionamiento interno de la aplicación, y de las situaciones clave a verificar, así como del tipo de datos que hay que verificar.
- b) Por el equipo analistas de la aplicación, siempre y cuando algunos de sus miembros estén implicados también en el proceso de implementación de la aplicación.
- c) Por personas totalmente independientes y distintas a las que han programado la aplicación, para evitar que aún de forma inconsciente, las pruebas se limiten a comprobar sólo los casos que ya han sido comprobados durante la implementación
- d) Por los programadores de la aplicación, aunque eventualmente puedan contar con la colaboración de algún analista y/o algún usuario de la aplicación.

Comprobar

Pasos para la resolución de un problema

5. Fase de Explotación y Mantenimiento.

Unidad Didáctica I

Fase de Explotación y Mantenimiento.

¿Y ahora qué? Se ha terminado el programa, el cliente da su visto bueno y nos vamos. ¿Qué hacemos con toda la documentación generada durante todo el proceso? ***Victor****, como siempre, no cree necesario guardar todo ese volumen de información, a pesar de lo que* ***María*** *le dijo sobre que nunca se sabe si la aplicación ha concluido.* ***José*** *le explica que una parte importante de todo el trabajo desarrollado es clasificarlo bien, porque en un momento dado puede ser necesario por múltiples motivos (puede que otro cliente necesite una aplicación similar, esta aplicación empiece a dar problemas en su funcionamiento, el cliente requiera mejoras, etc.)*

Una vez que la aplicación ya ha sido probada con éxito es el momento de instalarla para ponerla en marcha. Esta es la **fase de explotación** de la aplicación, y suele ser la más larga en el tiempo, y la que mayor coste supone a lo largo de la vida útil de la aplicación. A lo largo de la misma, será necesario ir haciendo algunas **tareas de mantenimiento de la aplicación**, para corregir errores, introducir mejoras, adaptaciones, etc.

En las grandes empresas de desarrollo del software, suele utilizarse un equipo de personas especializado en esta fase que además recoge todos los problemas encontrados en un documento de evaluación de la aplicación. Una vez concluida esta fase comprobando que la aplicación hace lo que debe sin errores y sin incoherencias es cuando se da por finalizado el servicio.

Pasos para la resolución de un problema

5.1. Ajustes.

Unidad Didáctica I

Ajustes

*Tras hacer la instalación de la aplicación en las oficinas de Servicios Locales, **José** y **Víctor** han hecho una demostración al personal encargado de utilizar esta aplicación y todos han quedado muy satisfechos, salvo una señora que ha detectado un "error" en la impresión de la nómina, ya que uno de los campos es norma de la empresa que se haga en color azul y esa información no se la habían notificado a los programadores. Así que toman nota y les invitan a que utilicen la aplicación, comprueben que se adapta a su trabajo y que cualquier nuevo fallo detectado lo anoten y se lo envíen para subsanarlo. **José** le dice a **Victor** que estos son los ajustes de última hora, que no son algo excepcional, sino que a veces es el cliente el que advierte con la nueva aplicación fallos o posibles mejoras en su propio funcionamiento.*

Incluso después de haber superado con éxito todas las pruebas realizadas, no es infrecuente que una vez que la aplicación está funcionando, aparezcan pequeños fallos en casos en los que a nadie se le ocurrió pensar. O incluso funcionamientos que inicialmente no se consideraron erróneos, pero que sí tienen esa consideración por parte de los usuarios de la aplicación una vez que la tienen operativa y requieren que el funcionamiento sea diferente al previsto.

Por ejemplo suele ocurrir que a la hora de imprimir la nómina de un trabajador, por el tratamiento manual de los documentos, se necesiten varias copias y así se incluye en los requerimientos hechos por el cliente. Pero ahora, una vez que tenemos la aplicación funcionando en red, cada uno de los departamentos implicados puede acceder al documento original e imprimirlo si lo necesita, así que no tiene sentido sacar varias copias de un documento si no es necesario y eso se va a decidir en cada departamento.

También es muy frecuente la distribución en el mercado de versiones Beta, para que los usuarios las prueben a fondo, descubriendo fallos ocultos, que deben ser corregidos.

Esto nos llevará a **replantear la aplicación desde la fase que sea necesaria**, y que normalmente será la codificación, pero que puede ser incluso el análisis, hasta obtener una nueva versión corregida y someterla de nuevo a un conjunto de pruebas específicas para probar el correcto funcionamiento donde antes fallaba.

Pasos para la resolución de un problema

5.2. Mejoras.

Unidad Didáctica I

Mejoras

*Durante la realización de la aplicación **José** ha tenido que plantear su trabajo según los requerimientos de la empresa, que ha intentado respetar al máximo. No obstante ha realizado una serie de sugerencias para mejorar el funcionamiento de la aplicación que considera que pueden ser aceptadas por los responsables de la empresa. Por ejemplo ha sugerido que sería interesante que en lugar de que todas las incidencias relacionadas con las nóminas de empleados se tramiten a través del departamento de personal, sería más operativo que cada sección enviara sus incidencias al ordenador central, incrementando las prestaciones. Pero esta sugerencia ha sido rechazada de momento. Una que sí ha sido admitida es que las nóminas puedan ser enviadas directamente al terminal del responsable de cada sección (incluso al terminal de cada trabajador) y sea éste el encargado de imprimirlas y entregarlas en mano a cada trabajador, evitando así algunos gastos en sobres y una persona haciendo el reparto.*

Aunque el funcionamiento de la aplicación sea correcto, y no se hayan detectado errores, habrá situaciones en las que se descubrirá que es posible realizar la misma tarea de forma distinta, de manera que se **aumente larapidez, o se abaraten costes**, (**mejoras en la eficiencia**) o se consiga que el manejo sea **más intuitivo y fácil o más ameno** para los usuarios. En todos estos casos, deberán realizarse los cambios oportunos en la aplicación, siguiendo para ello el mismo planteamiento de revisión de todas las etapas anteriores que sean necesarias, incluidas las pruebas.

Pasos para la resolución de un problema

5.3. Adaptaciones.

Unidad Didáctica I

Adaptaciones

*Una vez que la aplicación está en marcha, **José** hace un pequeño seguimiento de la misma durante las primeras semanas, resolviendo algunas dudas de los trabajadores y anotando todo aquello que considera de interés para posteriores modificaciones o para futuras aplicaciones. Sabe que probablemente, tarde o temprano la empresa necesite adaptar esta aplicación a cambios inevitables por causas muy diversas, lo importante es tener una documentación completa y clara para hacer más fácil esas adaptaciones.*

Con el paso del tiempo, es frecuente que surjan **situaciones nuevas, no previstas inicialmente, que requieran incluir funcionalidad nueva en nuestra aplicación**, o que se produzcan cambios normativos que obliguen a que la aplicación se adapte a los mismos modificando aquellas partes que se vean afectadas por los cambios. Incluso esos cambios pueden venir impuestos más por gustos que por exigencias objetivas reales, por ejemplo casos en los que se prefiere que la nómina se haga sobre un documento preimpreso de forma que la aplicación complete espacios reservados de un documento previamente diseñado y no imprima todo el contenido del mismo. Esto puede suponer un ahorro de tinta o tóner de impresora importante, y si el documento tiene un formato menor, ahorrará papel.

En todos los casos, si se decide adaptar la aplicación habrá que replantear las partes de las fases anteriores (normalmente desde el análisis).

AUTOEVALUACIÓN:

La fase de explotación y mantenimiento de la aplicación:

- a) Incluye ajustes y corrección de pequeños errores, mejoras en la eficiencia y usabilidad de la aplicación y adaptaciones para incluir nueva funcionalidad o adaptarla a nuevas exigencias
- b) Se termina con la comprobación mediante el test de pruebas correspondiente de que los cambios o mejoras introducidos cumplen con los requisitos marcados.
- c) Se termina con la depuración del código correspondiente a los cambios y mejoras introducidos en la aplicación.
- d) Ninguna de las respuestas anteriores es correcta.

Comprueba

Cualquier error detectado durante la fase de explotación de una aplicación nos llevará necesariamente a revisar:

- a) El análisis de la solución.
- b) El diseño del algoritmo.
- c) La codificación y pruebas del programa.
- d) Cualquiera de las fases previas, incluso todas, dependiendo de la gravedad del error

Comprobar

Pasos para la resolución de un problema