Introducción a la Ingeniería de Sistemas e Informática

Guía de Laboratorio

Visión

Ser la mejor organización de educación superior posible para unir personas e ideas que buscan hacer realidad sueños y aspiraciones de prosperidad en un entorno incierto

Misión

Somos una organización de educación superior que conecta personas e ideas para impulsar la innovación y el bienestar integral a través de una cultura de pensamiento y acción emprendedora.

Índice

Caso 1:

Responsabilidad social

Sección: ……..……..	Apellidos : ……..……
Docente :	Nombres : ……
Unidad: I	

Instrucciones: Leer el texto que se muestra a continuación y luego responder las preguntas del ítem II. Un miembro del grupo expone la respuesta.

I. Descripción o presentación del caso

Los 7 principios de responsabilidad social según la ISO 26000

1. Rendición de cuentas

La capacidad de una organización de acostumbrase a dar información sobre los impactos sociales, económicos y ambientales de su operación, que conlleva evidentemente no sólo positivos, sino también negativos y por ende asumir la responsabilidad de generar las medidas correctivas y establecer los candados necesarios para mitigarlos o evitarlos. Rendir cuentas debería también implicar aceptar la vigilancia pública y promover la respuesta a dicho escrutinio.

2. Transparencia

Esta práctica implica la costumbre de una organización de ser transparente en aquellas acciones que pueden vulnerar a la sociedad y al ambiente y por lo cual deberían brindar toda la información que requieran las partes interesadas en un lenguaje sencillo y en formatos accesibles.

3. Comportamiento Ético

La forma en la que una organización es lo que es, hace lo que hace y decide lo que decide, está directamente vinculada a los criterios de honestidad, equidad e integridad que establece la gobernanza organizacional para conducirse. Una estructura ética que moldee las decisiones y acciones, permite establecer que el fin no justifica los medios y por lo tanto que maximizar las ganancias debe conversar con la capacidad de incrementar los impactos positivos, y minimizar los negativos, en su entorno social y medioambiental.

4. Respeto a los intereses de las partes interesadas

Las organizaciones deben entender que los intereses de las partes involucradas son legítimos y deben ser atendidos y respetados, así sean distintos a los propios. Si bien es cierto, los objetivos de una empresa responden a sus dueños, existe un conjunto de partes interesadas que se ven afectadas por las actividades, acciones y decisiones, por lo que tomar en cuenta dichas expectativas es condición básica para operar de manera legítima y asegurar el éxito en el mediano y largo plazo. Porque valgan verdades, las partes interesadas tienen también y, felizmente, un potencial enorme de afectar una operación.

5. Respeto al principio de legalidad

Es el reconocimiento básico que ningún individuo y/u organización están por encima de la ley y, por ende, no tienen la potestad de actuar por fuera de sus marcos. Así, todas las instituciones deben cumplir las leyes y regulaciones aplicables, también en materia de responsabilidad social.

6. Respeto a la normatividad internacional de comportamiento

Una organización debería respetar la normativa internacional de comportamiento, a la vez que acatar el principio de respeto a la ley. Hoy los negocios son, sobre todo, globales, y con las cadenas de valor extendidas, es muy frecuente que subcontraten servicios en países donde las leyes laborales son bastante más laxas, por lo que la vulneración de derechos también es más plausible. Por ende, una organización debería contemplar respetar la normativa más exigente aún cuando la normativa nacional a la que esté sujeta no contemple las salvaguardas sociales y medioambientales.

7. Respeto a los derechos humanos

Una organización debería respetar los derechos humanos reconociendo, tanto su importancia y su universalidad. Es decir, son aplicables a todos los individuos en todos los países y culturas. Sobre todo allí donde ya sea por un vacío legal, o por prácticas inadecuadas, pueden ser vulnerados, la organización debería hacer un esfuerzo adicional por velar por ellos, respetarlos y protegerlos.

II. Consignas o preguntas reflexivas o actividades de resolución

2.1. Usted es Ingeniero de Sistemas. ¿Qué debe hacer para poner en práctica cada principio en su vida profesional?

Referencias bibliográficas consultadas y/o enlaces recomendados

- Piensaprofuturo. La súper ISO 26000 parte II: los 7 principios de responsabilidad social [en línea]. Disponible en http://www.piensaprofuturo.com/articulo/la-super-iso-26000-parte-ii-los-7-principios-de-responsabilidad-social-66

[Fecha de consulta: 01/02/2018].

Caso 2:

Suma en el sistema hexadecimal

Sección: ……..……..	Apellidos : ……..……
Docente :	Nombres : ……
Unidad: I	

Instrucciones: Leer el texto que se muestra a continuación y luego responder las preguntas del ítem II. Varios estudiantes elegidos al azar resuelven el problema en la pizarra.

I. Descripción o presentación del caso

Suma en el sistema hexadecimal

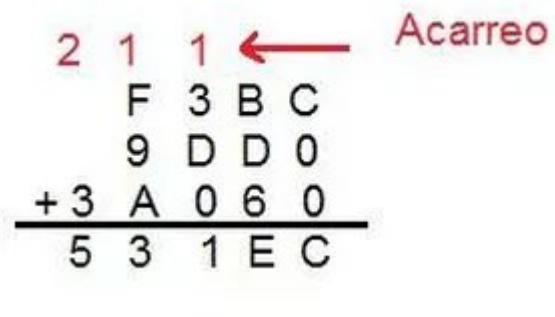
En el sistema hexadecimal los números se representan con dieciséis símbolos: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E y F. Se utilizan los caracteres A, B, C, D, E y F representando las cantidades decima-les 10, 11, 12, 13, 14 y 15 respectivamente. El valor de cada uno de estos símbolos depende, como es lógico, de su posición, que se calcula mediante potencias de base 16.

Suma hexadecimal: Cumple los mismos requisitos que la suma octal, la única diferencia es la base del sistema que se resta.

1. Se empieza a sumar de derecha a izquierda.

2. Sumar dígitos que se encuentran en la primera columna y se coloca el resultado debajo de la columna.

3. En caso de que la suma exceda la base del sistema, se escribe el resultado y se le restan 16, se coloca un acarreo en la siguiente columna, el valor del acarreo depende de las veces que haya superado la base del sistema y el valor que se obtiene de la resta se coloca debajo de la columna.



II. Consignas o preguntas reflexivas o actividades de resolución

II.1. Sumar en el sistema hexadecimal:

ABCD + 1A32

II.2. Sumar en el sistema hexadecimal:

123B + 224E

II.3. Sumar en el sistema hexadecimal:

111F + CCCC

II.4. Sumar en el sistema hexadecimal:

EFF1 + 3214

Referencias bibliográficas consultadas y/o enlaces recomendados

- tecelecuniminuto1. Suma y Resta en los Sistemas de Numeración [en línea]. Disponible en http://tecelecuniminuto1.wixsite.com/sistemasdenumeracion/suma-y-resta-en-el-sistema-hexadecimal

[Fecha de consulta: 01/02/2018].

Caso 3:

Representación de imágenes

Sección: ……..……..	Apellidos : ……..……
Docente :	Nombres : ……
Unidad: I	

Instrucciones: Leer el texto que se muestra a continuación y luego responder las preguntas del ítem II. Varios estudiantes elegidos al azar resuelven el problema en la pizarra.

I. Descripción o presentación del caso

Representación de imágenes

IMÁGENES

Hoy día las **imágenes** se representan en una computadora mediante uno de dos métodos: **gráficos de mapa de bits** o **gráficos de vectores** (figura 2.6).

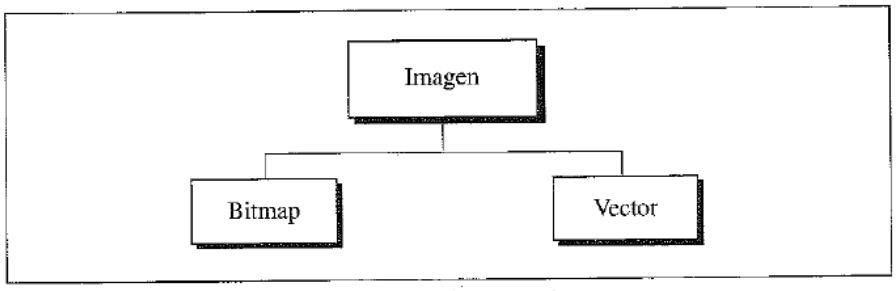


Figura 2.6 Métodos de representación de imágenes

Gráficos de mapa de bits

En este método, una imagen se divide en una matriz de **pixeles** (*picture elements:* **elementos de imagen**), donde cada pixel es un pequeño punto. El tamaño del pixel depende de lo que se conoce como *resolución*. Por ejemplo, una imagen puede dividirse en 1000 pixeles o 10 000 pixeles. En el segundo caso, aunque hay una mejor representación de la imagen (mejor resolución), se necesita más memoria para almacenarla.

Después de dividir una imagen en pixeles, a cada pixel se asigna un patrón de bits. El tamaño y el valor del patrón dependen de la imagen. Para una imagen formada sólo por puntos blancos y negros (por ejemplo, un tablero de ajedrez), un patrón de un bit es suficiente para representar un pixel. Un patrón de 0 representa un pixel negro y uno de 1 representa un pixel blanco. Luego los patrones se registran uno tras otro y se almacenan en la computadora. La figura 2.7 muestra una imagen de este tipo y su representación.

Si una imagen no se forma de pixeles puramente blancos y pixeles puramente negros, usted puede aumentar el tamaño del patrón de bits para representar escalas de grises. Por ejemplo, para mostrar cuatro niveles de la escala de grises, se puede usar un patrón de dos bits. Un pixel negro puede representarse por 00, un gris oscuro por 01, un pixel gris claro por 10 y un pixel blanco por 11.

Para representar imágenes a color, cada pixel coloreado se descompone en tres colores primarios: rojo, verde y azul (RGB). Luego se mide la intensidad de cada color y se le asigna un

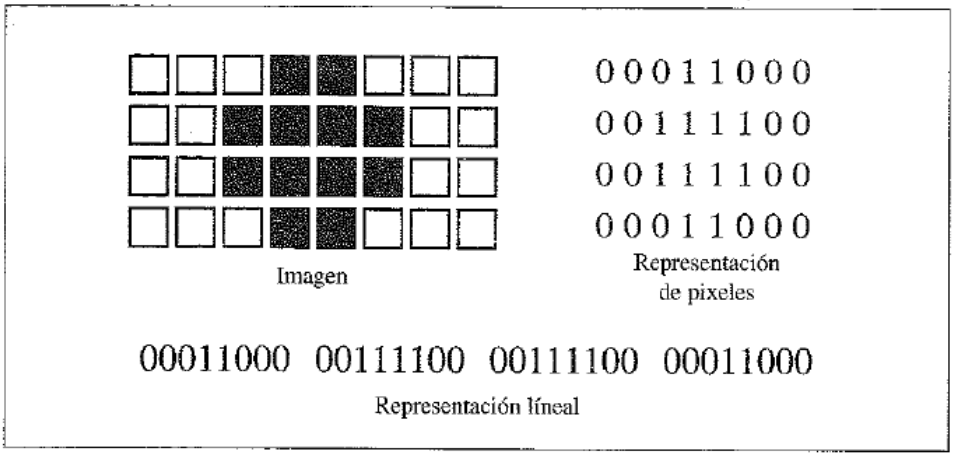


Figura 2.7 Método de gráficos de mapa de bits de una imagen blanca y negra

patrón de bits (por lo general ocho bits). En otras palabras, cada pixel tiene tres patrones de bits: uno para representar la intensidad del color rojo, uno para la intensidad del color verde y uno para la intensidad del color azul. Por ejemplo, la figura 2.8 muestra cuatro patrones de bits para algunos pixeles en una imagen a color.

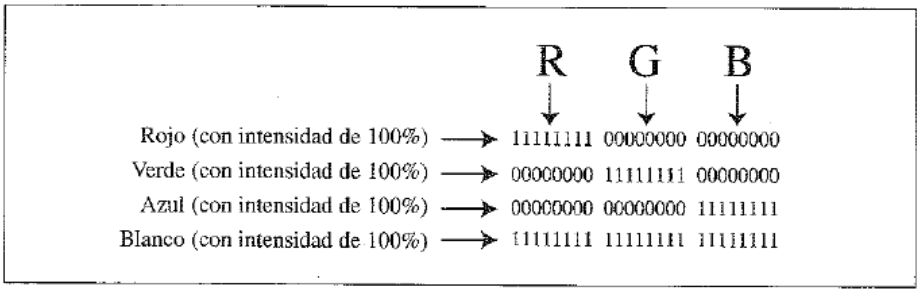


Figura 2.8 Representación de pixeles de color

Gráficos de vectores

El problema con el método de los gráficos de mapa de bits es que los patrones de bits exactos para representar una imagen particular deben guardarse en una computadora. Posteriormente, si usted desea cambiar el tamaño de la imagen debe cambiar el tamaño de los pixeles, lo cual crea una apariencia difusa y granulada. No obstante, el método de gráficos de vector no guarda los patrones de bits. Una imagen se descompone en una combinación de curvas y líneas. Cada curva o línea se representa por medio de una fórmula matemática. Por ejemplo, una línea puede describirse mediante las coordenadas de sus puntos extremos y un círculo puede describirse mediante las coordenadas de su centro y la longitud de su radio. La combinación de estas fórmulas se almacena en una computadora. Cuando la imagen se va a desplegar o imprimir, el tamaño de la imagen se proporciona al sistema como una entrada. El sistema rediseña la imagen con el nuevo tamaño y usa la misma fórmula para dibujar la imagen. En este caso, cada vez que una imagen se dibuja, la fórmula se vuelve a evaluar.

II. Consignas o preguntas reflexivas o actividades de resolución

II.1. Representar la letra A utilizando mapa de bits con una matriz de 5x5.

II.2. Representar el digito 4 utilizando mapa de bits con una matriz de 5x5.

II.3. Representar la letra C utilizando mapa de bits con una matriz de 5x5.

II.4. Representar la letra 7 utilizando mapa de bits con una matriz de 5x5.

Referencias bibliográficas consultadas y/o enlaces recomendados

- Forouzan B (2003). Introducción a la ciencia de la computación. México D.F: International Thomson Editores.

Caso 4:

Arquitectura Von Neumann y arquitectura Harvard

Sección: ……..……..	Apellidos : ……..……..
Docente :	Nombres : ……
Unidad: II	

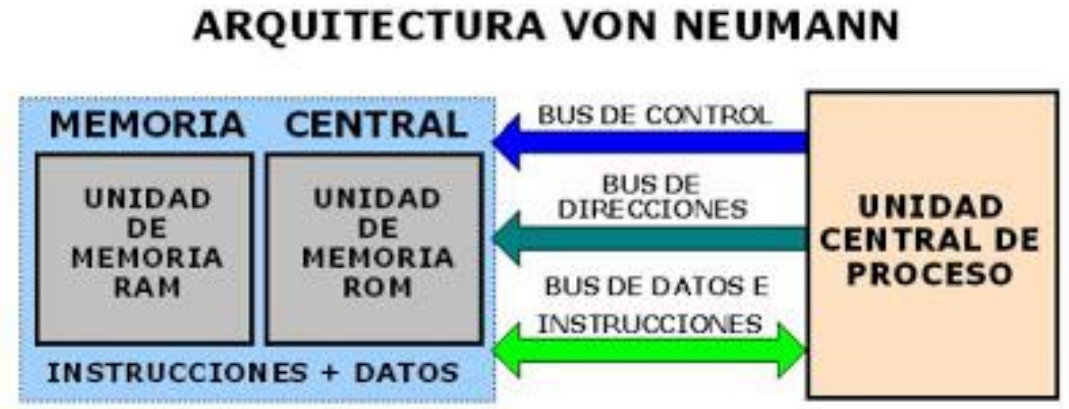
Instrucciones: Leer el texto que se muestra a continuación y luego responder las preguntas del ítem II. Varios estudiantes elegidos al azar resuelven el problema en la pizarra.

I. Descripción o presentación del caso

Arquitectura Von Neumann y arquitectura Harvard

Hay dos arquitecturas distintas relacionadas con el uso y distribución de la memoria: **Arquitectura de Von Neumman** y **Arquitectura Harvard.**

Arquitectura de Von Neumann: Tradicionalmente los sistemas con microprocesadores se basan en esta arquitectura, en la cual la unidad central de proceso (CPU), está conectada a una memoria principal única (casi siempre sólo RAM) donde se guardan las instrucciones del programa y los datos. A dicha memoria se accede a través de un sistema de buses único (control, direcciones y datos).



En un sistema con **arquitectura Von Neumann** el tamaño de la unidad de datos o instrucciones está fijado por el ancho del bus que comunica la memoria con la CPU. Así un microprocesador de 8 bits con un bus de

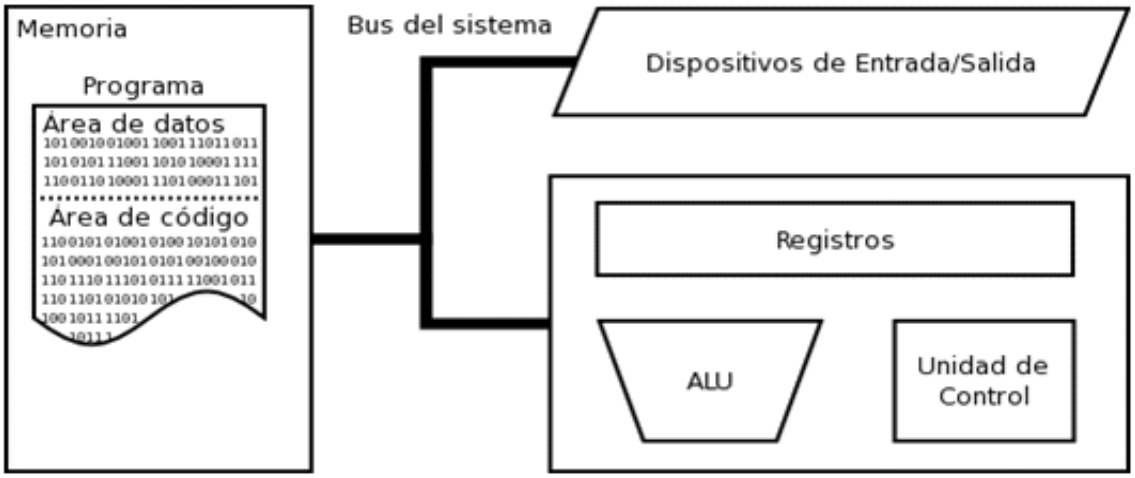
8 bits, tendrá que manejar datos e instrucciones de una o más unidades de 8 bits (bytes) de longitud. Si tiene que acceder a una instrucción o dato de más de un byte de longitud, tendrá que realizar más de un acceso a la memoria.

El tener un único bus hace que el microprocesador sea más lento en su respuesta, ya que no puede buscar en memoria una nueva instrucción mientras no finalicen las transferencias de datos de la instrucción anterior.

Las principales limitaciones que nos encontramos con la **arquitectura Von Neumann** son:

- La limitación de la longitud de las instrucciones por el bus de datos, que hace que el microprocesador tenga que realizar varios accesos a memoria para buscar instrucciones complejas.
- La limitación de la velocidad de operación a causa del bus único para datos e instrucciones que no deja acceder simultáneamente a unos y otras, lo cual impide superponer ambos tiempos de acceso

Los ordenadores con **arquitectura Von Neumann** constan de las siguientes partes:



La **arquitectura Von Neumann** realiza o emula los siguientes pasos secuencialmente:

1) Obtiene la siguiente instrucción desde la memoria en la dirección indicada por el contador de programa y la guarda en el registro de instrucción.

2) Aumenta el contador de programa en la longitud de la instrucción para apuntar a la siguiente.

3) Descodifica la instrucción mediante la unidad de control. Ésta se encarga de coordinar el resto de componentes del ordenador para realizar una función determinada.

4) Se ejecuta la instrucción. Ésta puede cambiar el valor del contador del programa, permitiendo así operaciones repetitivas.

5) Regresa al paso N° 1.

Conclusión:

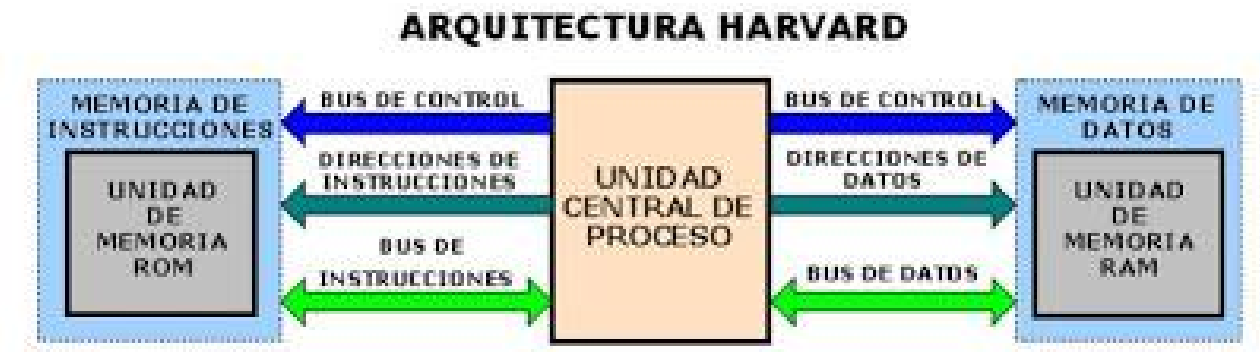
* La mayoría de las computadoras todavía utilizan la arquitectura Von Neumann, propuesta a principios de los años 40 por John Von Neumann.

* La arquitectura Von Neumann describe a la computadora con 4 secciones principales: la unidad lógica y aritmética (ALU), la unidad de control, la memoria, y los dispositivos de entrada y salida (E/S).

* En este sistema, la memoria es una secuencia de celdas de almacenamiento numeradas, donde cada una es un bit, o unidad de información. La instrucción es la información necesaria para realizar, lo que se desea, con la computadora. Las celdas contienen datos que se necesitan para llevar a cabo las instrucciones, con la computadora.

* El tamaño de cada celda y el número de celdas varía mucho de computadora a computadora, y las tecnologías empleadas para la memoria han cambiado bastante; van desde los relés electromecánicos, tubos llenos de mercurio en los que se formaban los pulsos acústicos, matrices de imanes permanentes, transistores individuales a circuitos integrados con millones de celdas en un solo chip.

Arquitectura Harvard: Este modelo, que utilizan los Microcontroladores PIC, tiene la unidad central de proceso (CPU) conectada a dos memorias (una con las instrucciones y otra con los datos) por medio de dos buses diferentes.



Una de las memorias contiene solamente las instrucciones del programa (Memoria de Programa), y la otra sólo almacena datos (Memoria de Datos).

Ambos buses son totalmente independientes lo que permite que la CPU pueda acceder de forma independiente y simultánea a la memoria de datos y a la de instrucciones. Como los buses son independientes estos pueden tener distintos contenidos en la misma dirección y también distinta longitud.

También la longitud de los datos y las instrucciones puede ser distinta, lo que optimiza el uso de la memoria en general.

Para un procesador de **Set de Instrucciones Reducido**, o **RISC (Reduced Instrucción Set Computer)**, el set de instrucciones y el bus de memoria de programa pueden diseñarse de tal manera que todas las instrucciones tengan una sola posición de memoria de programa de longitud.

Además, al ser los buses independientes, la CPU puede acceder a los datos para completar la ejecución de una instrucción, y al mismo tiempo leer la siguiente instrucción a ejecutar.

Ventajas de esta arquitectura:

* El tamaño de las instrucciones no esta relacionado con el de los datos, y por lo tanto puede ser optimizado para que cualquier instrucción ocupe una sola posición de memoria de programa, logrando así mayor velocidad y menor longitud de programa.

* El tiempo de acceso a las instrucciones puede superponerse con el de los datos, logrando una mayor velocidad en cada operación.

II. Consignas o preguntas reflexivas o actividades de resolución

II.1. Realizar un mapa conceptual que permita resumir la lectura anterior.

Referencias bibliográficas consultadas y/o enlaces recomendados

- rcmcomputointegrado. Arquitectura Von Neumann y arquitectura Harvard [en línea]. Disponible en http://rcmcomputointegrado.blogspot.com/2012/04/arquitectura-von-neumann.html, [Fecha de consulta: 01/02/2018].

Caso 5:

Arquitectura Android

Sección: ……..……..	Apellidos : ……..……
Docente :	Nombres : ……
Unidad: II	

Instrucciones: Leer el texto que se muestra a continuación y luego responder las preguntas del ítem II. Varios estudiantes elegidos al azar resuelven el problema en la pizarra.

I. Descripción o presentación del caso

ARQUITECTURA ANDROID

Como ya se ha mencionado, Android es una plataforma para dispositivos móviles que contiene una pila de software donde se incluye un sistema operativo, middleware **y aplicaciones básicas para el usuario.**

En las siguientes líneas se dará una visión global por capas de cuál es la arquitectura empleada en Android. Cada una de estas capas utiliza servicios ofrecidos por las anteriores, y ofrece a su vez los suyos propios a las capas de niveles superiores, tal como muestra la siguiente figura ((c) Google):

- **Aplicaciones:** Este nivel contiene, tanto las incluidas por defecto de Android como aquellas que el usuario vaya añadiendo posteriormente, ya sean de terceras empresas o de su propio desarrollo. Todas estas aplicaciones utilizan los servicios, las API y librerías de los niveles anteriores.
- **Framework de Aplicaciones:** Representa fundamentalmente el conjunto de herramientas de desarrollo de cualquier aplicación. Toda aplicación que se desarrolle para Android, ya sean las propias del dispositivo, las desarrolladas por Google o terceras compañías, o incluso las que el propio usuario cree, utilizan el mismo conjunto de API y el mismo "framework", representado por este nivel.

Entre las API más importantes ubicadas aquí, se pueden encontrar las siguientes:

- *Activity Manager:* Conjunto de API que gestiona el ciclo de vida de las aplicaciones en Android.
- *Window Manager:* Gestiona las ventanas de las aplicaciones y utiliza la librería Surface Manager.
- *Telephone Manager:* Incluye todas las API vinculadas a las funcionalidades propias del teléfono (llamadas, mensajes, etc.).
- *Content Provider:* Permite a cualquier aplicación compartir sus datos con las demás aplicaciones de Android. Por ejemplo, gracias a esta API la información de contactos, agenda, mensajes, etc. será accesible para otras aplicaciones.
- *View System:* Proporciona un gran número de elementos para poder construir interfaces de usuario (GUI), como listas, mosaicos, botones, "check-boxes", tamaño de ventanas, control de las interfaces mediante teclado, etc. Incluye también algunas vistas estándar para las funcionalidades más frecuentes.

••••••• · *Location Manager:* Posibilita a las aplicaciones la obtención de información de localización y posicionamiento.

· *Notification Manager:* Mediante el cual las aplicaciones, usando un mismo formato, comunican al usuario eventos que ocurran durante su ejecución: una llamada entrante, un mensaje recibido, conexión Wi-Fi disponible, ubicación en un punto determinado, etc. Si llevan asociada alguna acción, en Android denominada **Intent**, (por ejemplo, atender una llamada recibida) ésta se activa mediante un simple clic.

· *XMPP Service:* Colección de API para utilizar este protocolo de intercambio de mensajes basado en XML.

- **Librerías:** La siguiente capa se corresponde con las librerías utilizadas por Android. Éstas han sido escritas utilizando C/C++ y proporcionan a Android la mayor parte de sus capacidades más características. Junto al núcleo basado en Linux, estas librerías constituyen el corazón de Android.

Entre las librerías más importantes ubicadas aquí, se pueden encontrar las siguientes:

· *Librería libc:* Incluye todas las cabeceras y funciones según el estándar del lenguaje C. Todas las demás librerías se definen en este lenguaje.

· *Librería Surface Manager:* Es la encargada de componer los diferentes elementos de navegación de pantalla. Gestiona también las ventanas pertenecientes a las distintas aplicaciones activas en cada momento.

· *OpenGL/SL y SGL:* Representan las librerías gráficas y, por tanto, sustentan la capacidad gráfica de Android. OpenGL/SL maneja gráficos en 3D y permite utilizar, en caso de que esté disponible en el propio dispositivo móvil, el hardware encargado de proporcionar gráficos 3D. Por otro lado, SGL proporciona gráficos en 2D, por lo que será la librería más habitualmente utilizada por la mayoría de las aplicaciones. Una característica importante de la capacidad gráfica de Android es que es posible desarrollar aplicaciones que combinen gráficos en 3D y 2D.

· *Librería Media Libraries:* Proporciona todos los códecs necesarios para el contenido multimedia soportado en Android (vídeo, audio, imágenes estáticas y animadas, etc.)

· *FreeType:* Permite trabajar de forma rápida y sencilla con distintos tipos de fuentes.

· *Librería SSL:* Posibilita la utilización de dicho protocolo para establecer comunicaciones seguras.

· *Librería SQLite:* Creación y gestión de bases de datos relacionales.

· *Librería WebKit:* Proporciona un motor para las aplicaciones de tipo navegador y forma el núcleo del actual navegador incluido por defecto en la plataforma Android.

- **Tiempo de ejecución de Android:** Al mismo nivel que las librerias de Android se sitúa el entorno de ejecución. Éste lo constituyen las Core Libraries, que son librerias con mulititud de clases Java y la máquina vistual Dalvik.

- **Núcleo Linux**: Android utiliza el **núcleo de Linux 2.6** como una capa de abstracción para el hardware disponible en los dispositivos móviles. Esta capa contiene los drivers necesarios para que cualquier componente hardware pueda ser utilizado mediante las llamadas correspondientes. Siempre que un fabricante incluye un nuevo elemento de hardware, lo primero que se debe realizar para que pueda ser utilizado desde Android es crear las librerias de control o drivers necesarios dentro de este kernel de Linux embebido en el propio Android.

II. Consignas o preguntas reflexivas o actividades de resolución

II.1. ¿Qué capa considera más importante y por qué?

Referencias bibliográficas consultadas y/o enlaces recomendados

- Software de Comunicaciones. Arquitectura Android [en línea]. Disponible en https://sites.google.com/site/swcuc3m/home/android/generalidades/2-2-arquitectura-de-android, [Fecha de consulta: 01/02/2018].

Caso 6:

Protocolos de red

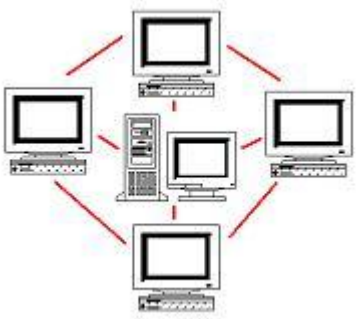
Sección: ……..……..	Apellidos : ……..……..
Docente :	Nombres : ……
Unidad: II	

Instrucciones: Leer el texto que se muestra a continuación y luego responder las preguntas del ítem II. Varios estudiantes elegidos al azar resuelven el problema en la pizarra.

I. Descripción o presentación del caso

PROTOCOLO DE RED

Protocolo es el término que se emplea para denominar al conjunto de normas, reglas y pautas que sirven para guiar una conducta o acción. **Red**, por su parte, es una clase de estructura o sistema que cuenta con un patrón determinado.



El concepto de **protocolo de red** se utiliza en el contexto de la **informática** para nombrar a las normativas y los criterios que fijan cómo deben comunicarse los diversos componentes de un cierto sistema de interconexión. Esto quiere decir que, a través de este protocolo, los dispositivos que se conectan en red pueden **intercambiar datos**.

También conocido como **protocolo de comunicación**, el protocolo de red establece la **semántica** y la **sintaxis** del intercambio de información, algo que constituye un estándar. Las computadoras en red, de este modo, tienen que actuar de acuerdo a los parámetros y los criterios establecidos por el protocolo en cuestión para lograr comunicarse entre sí y para recuperar datos que, por algún motivo, no hayan llegado a destino.

En el protocolo de red se incluyen diversas informaciones que son imprescindibles para la **conexión**. El protocolo indica cómo se concreta la conexión física, establece la manera en que debe comenzar y terminar la comunicación, determina cómo actuar ante datos corrompidos, protege la información ante el ataque de intrusos, señala el eventual cierre de la transmisión, etc.

Existen protocolos de red en cada **capa** o **nivel** de la conexión. La capa inferior refiere a la conectividad física que permite el desarrollo de la red (con **cables UTP**, **ondas de radio**, etc.), mientras que la capa más avanzada está vinculada a las aplicaciones que utiliza el usuario de la **computadora** (con protocolos como **HTTP**, **FTP**, **SMTP**, **POP** y otros).

HTTP

El *Protocolo de Transferencia de Hipertexto*se usa en todas las **transacciones** que tienen lugar en Internet, ya que cuenta con la definición de la semántica y la sintaxis que deben usar los servidores, los proxies y los clientes (todos componentes de la arquitectura web) para entablar una comunicación entre ellos. Se trata de un protocolo que se orienta a la transacción y se apoya en el esquema "petición-respuesta", típico entre un cliente (también se denomina *agente del usuario* y puede ser, por ejemplo, un navegador de Internet) y un servidor. La información que se transmite en este proceso recibe el nombre de **recurso**, identificado a través de un URL (*Localizador Uniforme de Recursos*).

FTP

El *Protocolo de Transferencia de Archivos*, por su parte, se utiliza cuando se desea enviar y recibir archivos de un sistema a otro, siempre que ambos se basen en la arquitectura *cliente-servidor* y que se encuentren conectados a una red que cumpla con el **TCP**, explicado en la definición de **protocolo de comunicación**. El FTP permite que un usuario se conecte a un servidor para bajar archivos o bien para subirlos, sin la necesidad de que ambos equipos utilicen el mismo sistema operativo.

SMTP

Con un nombre menos conocido que los dos anteriores, el *Protocolo para transferencia simple de correo* es utilizado una cantidad incalculable de veces al día por usuarios de todo el mundo, ya que da forma al intercambio de mensajes de **correo electrónico** (también conocido como *e-mail* o *email*) entre una amplia gama de dispositivos, como ser los teléfonos móviles, las tabletas y los ordenadores. Se trata de un estándar oficial cuya operación se encuentra en manos de los proveedores de servicios de email.

POP

El *Protocolo de Oficina de Correo* o *de Oficina Postal* brinda a los usuarios la posibilidad de recibir y almacenar el correo electrónico en un **equipo** local. En la actualidad se prefiere el uso de POP3, la versión más reciente, dado que las primeras dos se consideran obsoletas.

II. Consignas o preguntas reflexivas o actividades de resolución

II.1. ¿Qué protocolo considera más importante y por qué?

Referencias bibliográficas consultadas y/o enlaces recomendados

- Definicion.de. Protocolo de red [en línea]. Disponible en https://definicion.de/protocolo-de-red/, [Fecha de consulta: 01/02/2018].

Caso 7:

Uso de Internet en el Perú

Sección: ……..……..	Apellidos : ……..……..
Docente :	Nombres : ……
Unidad: II	

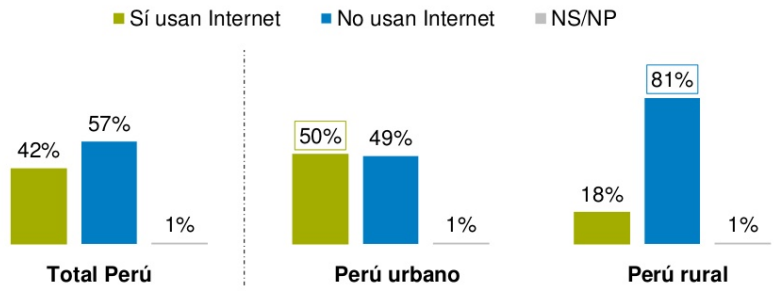
Instrucciones: Leer el texto que se muestra a continuación y luego responder las preguntas del ítem II. Varios estudiantes elegidos al azar resuelven el problema en la pizarra.

I. Descripción o presentación del caso

USO DE INTERNET EN EL PERÚ – OCTUBRE 2015

Uso de Internet en el Perú

El número de usuarios de Internet en el Perú llega al 42%. Esta cifra se incrementa cuando se considera únicamente el ámbito urbano (50%) y se reduce considerablemente en el entorno rural, donde la conectividad llega a sólo 18%.

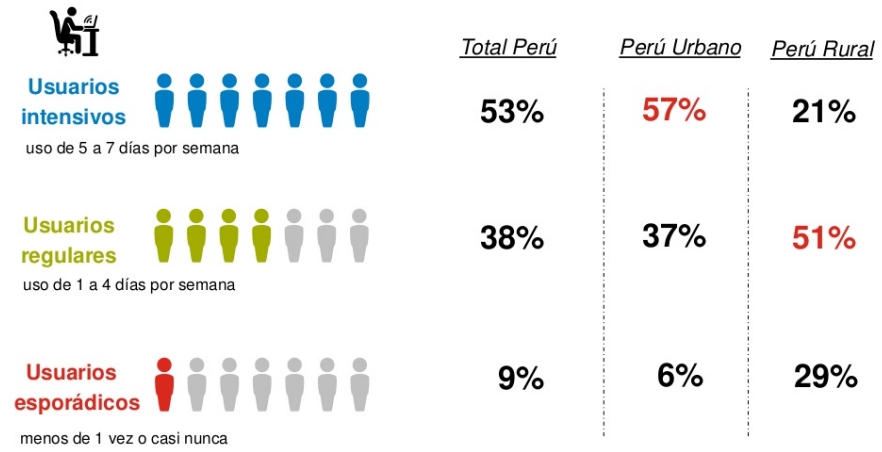


	Total	Nivel Socioeconómico			Sexo		Región		Edad		
		A/B	C	D/E	Hombres	Mujeres	Lima	Interior	18-24	25-39	40 a más
Usan Internet	42%	**78%**	**58%**	26%	**44%**	40%	**57%**	34%	**70%**	**56%**	24%
No usan Internet	57%	20%	**42%**	**73%**	55%	**59%**	43%	**65%**	30%	**43%**	**74%**
NS/NP	1%	2%	-	1%	1%	1%	-	1%	-	1%	2%

Base octubre 2015: (100%) Total de entrevistados - Nacional urbano rural (1308) 6

Perfil del usuario de Internet- Urbano vs Rural

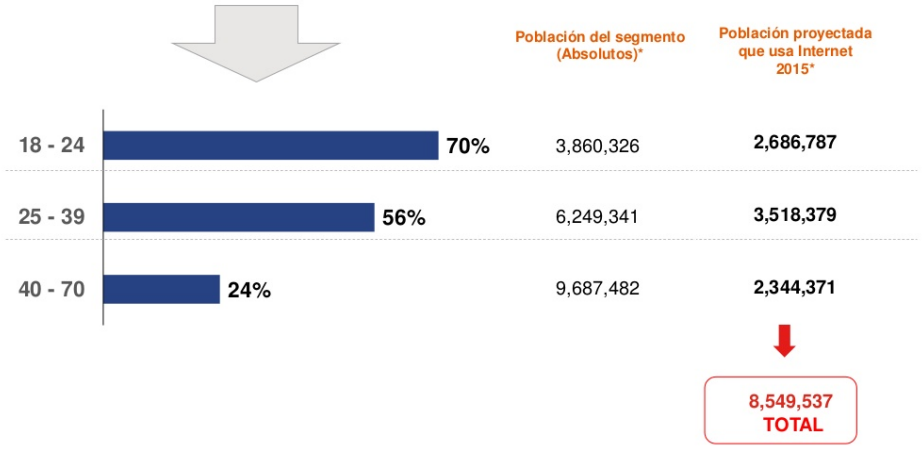
Cuando se comparan los dos ámbitos, la zona urbana congrega un mayor número de usuarios intensivos, mientras que en el ámbito rural todavía existe un importante número de usuarios regulares (51%) y esporádicos (29%).



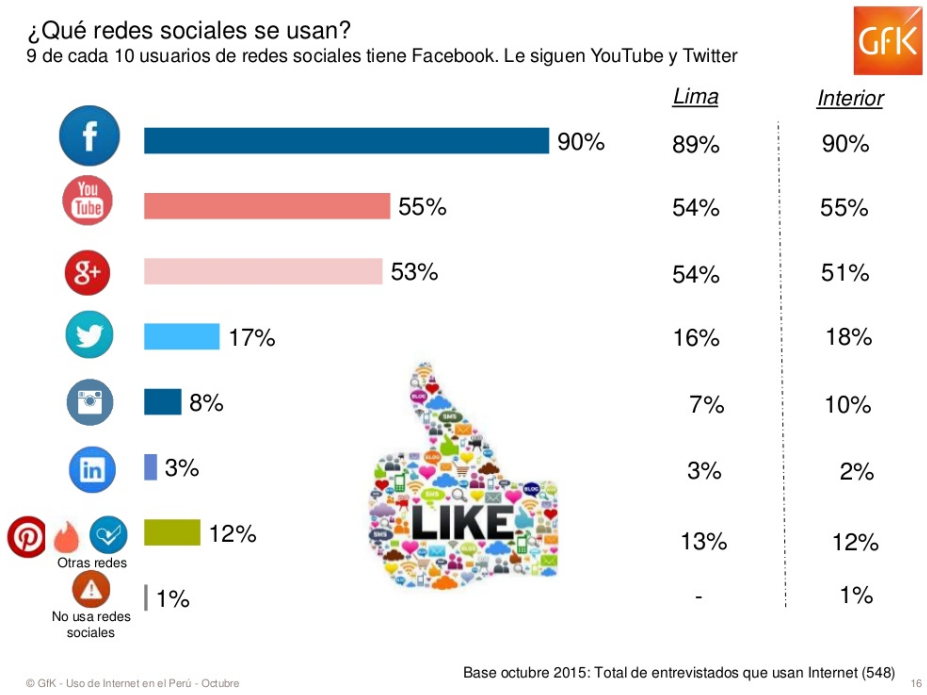
Base octubre 2015: Total de entrevistados que usan internet- Nacional urbano rural (548) 8

Uso de Internet de cada grupo de edad

El segmento más joven (18-24) es el que más utiliza Internet, representando al 31% de los peruanos mayores de edad que utilizan este servicio.



*Según APEIM 2015, Perú urbano y rural. Población en el rango de 18 a 70 años
Base octubre 2015: (100%) Total de entrevistados - Nacional urbano rural (1308) 12



II. Consignas o preguntas reflexivas o actividades de resolución

II.1. Analizar las estadísticas anteriores.

II.2. Con cada integrante del aula realice una encuesta y graficar el resultado a la pregunta ¿Para qué usa internet? Cada integrante debe dar 2 respuestas.

II.3. Con cada integrante del aula realice una encuesta y graficar el resultado a la pregunta ¿Qué tipos de vídeo visualiza en youtube? Cada integrante debe dar 2 respuestas.

Referencias bibliográficas consultadas y/o enlaces recomendados

- GFK Perú (2015). Uso de internet en el Perú [en línea]. Disponible en https://es.slideshare.net/GfKPeru/gfk-per-uso-de-internet-en-el-per-octubre-2015, [Fecha de consulta: 01/02/2018].

Caso 8:

Diagramas de flujo

Sección: ……..……..	Apellidos : ……..……
Docente :	Nombres : ……
Unidad: III	

Instrucciones: Leer el texto que se muestra a continuación y luego responder las preguntas del ítem II. Varios estudiantes elegidos al azar resuelven el problema en la pizarra.

I. Descripción o presentación del caso

En la figura 1.1 podemos observar las etapas que debemos seguir para solucionar algún problema.

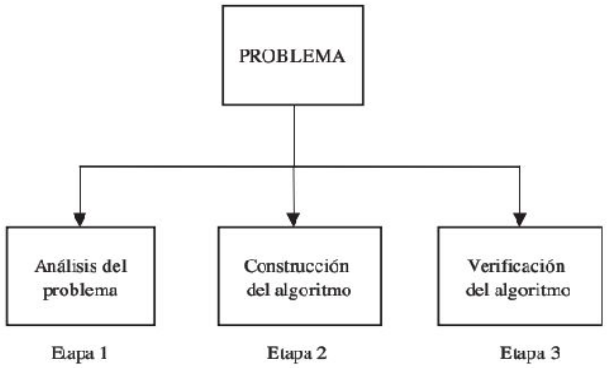


FIGURA 1.1
Etapas para solucionar un problema.

Por otra parte, las características que deben tener los algoritmos son las siguientes:

Precisión: Los pasos a seguir en el algoritmo se deben *precisar* claramente.

Determinismo: El algoritmo, dado un conjunto de datos de entrada idéntico, siempre debe arrojar los mismos resultados.

Finitud: El algoritmo, independientemente de la complejidad del mismo, siempre debe tener longitud finita.

El algoritmo consta de tres secciones o módulos principales (figura 1.2).

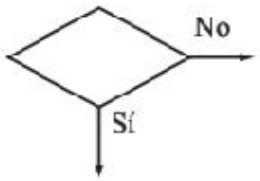
FIGURA 1.2
Módulos o secciones de un algoritmo.

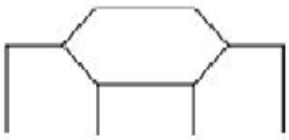
1.2 Diagramas de flujo

El diagrama de flujo representa la esquematización gráfica de un algoritmo. En realidad muestra gráficamente los pasos o procesos a seguir para alcanzar la solución de un problema. La construcción correcta del mismo es muy importante, ya que a partir de éste se escribe el programa en un lenguaje de programación determinado. En este caso utilizaremos el lenguaje C, aunque cabe recordar que el diagrama de flujo se debe construir de manera independiente al lenguaje de programación. El diagrama de flujo representa la solución del problema. El programa representa la implementación en un lenguaje de programación.

A continuación, en la tabla 1.1 se presentan los símbolos que se utilizarán, junto con una explicación de los mismos. Éstos satisfacen las recomendaciones de la *International Organization for Standardization* (ISO) y el *American National Standards Institute* (ANSI).

TABLA 1.1. Símbolos utilizados en los diagramas de flujo

Representación del símbolo	*Explicación del símbolo*
	Se utiliza para marcar el *inicio* y el *fin* del diagrama de flujo.
	Se utiliza para introducir los datos de entrada. Expresa *lectura*.
	Representa un *proceso*. En su interior se colocan asignaciones, operaciones aritméticas, cambios de valor de celdas en memoria, etc.
	Se utiliza para representar una *decisión*. En su interior se almacena una condición, y, dependiendo del resultado, se sigue por una de las ramas o caminos alternativos. Este símbolo se utiliza con pequeñas variaciones en las estructuras selectivas `if` e `if-else` que estudiaremos en el siguiente capítulo, así como en las estructuras repetitivas `for`, `while` y `do-while`, que analizaremos en el capítulo 3.



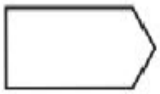
Se utiliza para representar una decisión múltiple, `switch`, que analizaremos en el siguiente capítulo. En su interior se almacena un selector, y, dependiendo del valor de dicho selector, se sigue por una de las ramas o caminos alternativos.

Se utiliza para representar la impresión de un resultado. Expresa *escritura*.

Expresan la dirección del flujo del diagrama.

Expresa conexión dentro de una misma página.

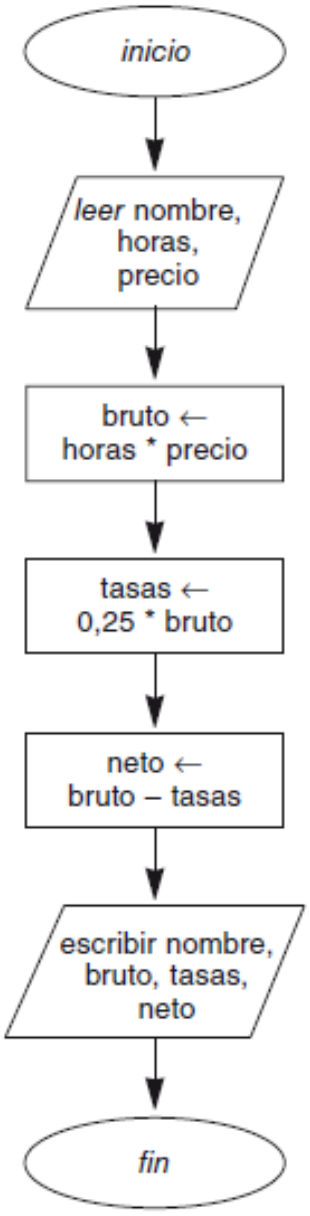
Representa conexión entre páginas diferentes.



Se utiliza para expresar un módulo de un problema, subproblema, que hay que resolver antes de continuar con el flujo normal del diagrama.

Problema:

Calcular el salario bruto y el salario neto de un trabajador "por horas" conociendo el nombre, número de horas trabajadas, impuestos a pagar y salario neto.



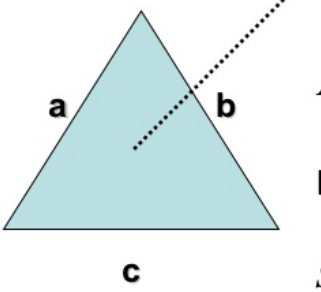
II. Consignas o preguntas reflexivas o actividades de resolución

II.1. Realizar el diagrama de flujo para un programa que permita calcular el área de un rectángulo.

II.2. Realizar el diagrama de flujo para un programa que permita calcular el área de un triángulo en función a sus lados. El área se calcula de acuerdo con la fórmula de Heron:

25

Fórmula de Herón para calcular el área de un triángulo si conocemos la medida de sus lados.



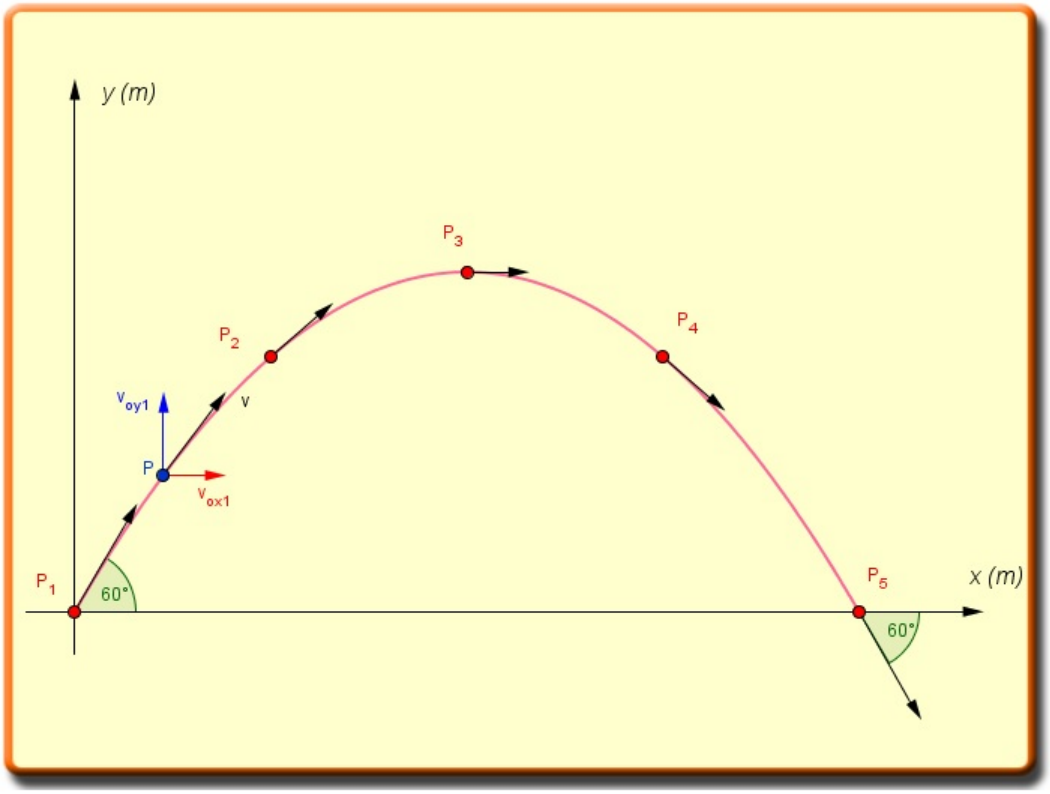
El área del triángulo está dada por la fórmula:

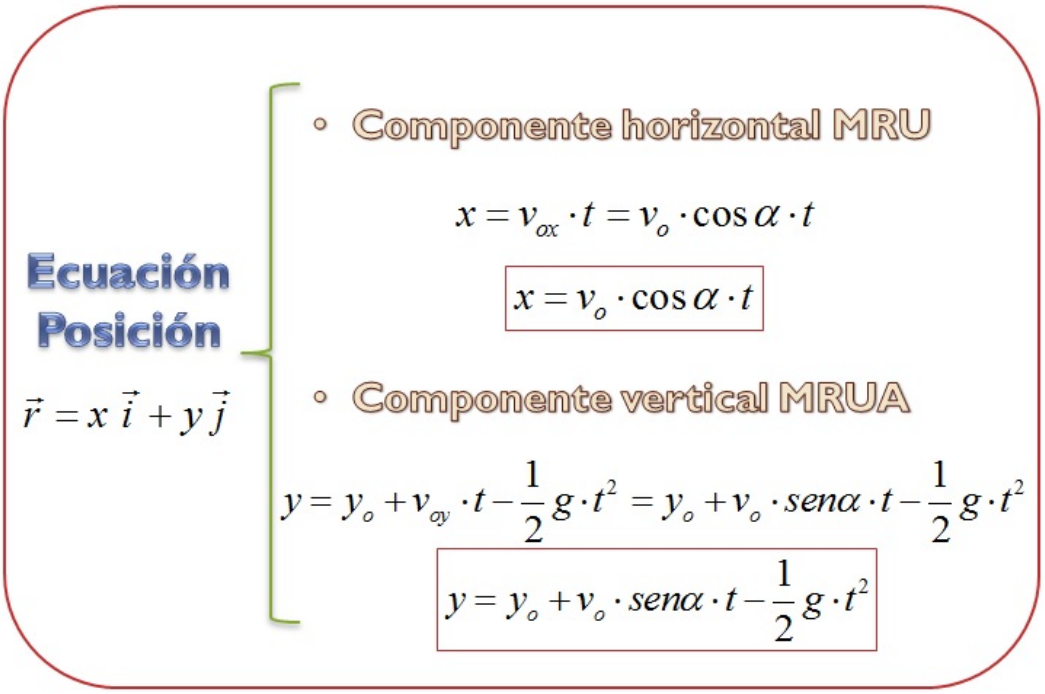
$$A = \sqrt{s(s-a)(s-b)(s-c)}$$

Donde;

$$s = \frac{1}{2}(a+b+c)$$

II.3. Realizar el diagrama de flujo para un programa que permita calcular la posición del movimiento parabólico. La posición se calcula de acuerdo con la siguiente fórmula:





Referencias bibliográficas consultadas y/o enlaces recomendados

- Joyanes L (2008). Fundamentos de programación. 4ta edición. Madrid: McGrawHill.
- Cairó O (2006). Fundamentos de programación: piensa en C. Madrid: Prentice Hall.

Caso 9:

Estructura condicional if en Python

Sección: ……..……..	Apellidos : ……..……
Docente :	Nombres : ……
Unidad: III	

Instrucciones: Leer el texto que se muestra a continuación y luego responder las preguntas del ítem II. Varios estudiantes elegidos al azar resuelven el problema en la pizarra.

I. Descripción o presentación del caso

A continuación, se presenta el análisis de la sentencia if.

4.1 if

Las condiciones sirven para seleccionar cuándo debe ser activada una parte del programa y cuándo debe ser simplemente ignorada. En Python, la estructura de decisión es el `if`. Su formato es presentado en la lista 4.1.

▶ **Lista 4.1 – Formato de la estructura de condicional if**

```
if <condición>:
    bloque verdadero
```

El `if` es nuestro "si". Podremos entonces entenderlo en español de la siguiente forma: si la condición es verdadera, haga algo.

Veamos un ejemplo presentado en la lista 4.2: leer dos valores e imprimir el mayor de ellos.

Un bloque de líneas en Python puede tener más de una línea, aunque el último ejemplo muestre solo dos bloques con una línea en cada uno. Si necesita dos o más en el mismo bloque, escriba esas líneas en la misma dirección o en la misma columna de la primera línea del bloque. Eso basta para representarlo.

Un problema común aparece cuando tenemos que pagar impuesto de renta. Normalmente pagamos el impuesto de renta por franja de salario. Imagine que para salarios menores de € 1.000,00 no tenemos que pagar impuesto, es decir que tenemos una alícuota de 0%. Para salarios entre € 1.000,00 y € 3.000,00 pagaríamos 20%. Encima de esos valores, la alícuota sería de 35%. Ese problema se parecería mucho al anterior si el impuesto no fuese cobrado de manera diferente para cada franja; quien gana € 4.000,00 tiene los primeros € 1.000,00 exentos de impuesto; para los valores entre € 1.000,00 y € 3.000,00 paga 20%, y para los valores restantes paga 35%. Veamos la solución en la lista del programa 4.4.

▶ Lista 4.4 – Cálculo del impuesto de renta

```
salario = float(input("Digite el salario para cálculo del impuesto: "))
base = salario ❶
impuesto = 0
if base > 3000: ❷
   impuesto = impuesto + ((base - 3000) * 0.35) ❸
   base = 3000 ❹
if base > 1000: ❺
   impuesto = impuesto + ((base - 1000) * 0.20) ❻
print("Salario: €%6.2f Impuesto a pagar: €%6.2f" % (salario, impuesto))
```


El programa de la lista 4.4 es muy interesante. Trate de ejecutarlo algunas veces y compare el valor impreso con el valor calculado por Ud. Rastree el programa y trate de entender qué hace antes de leer el párrafo siguiente. Verifique qué sucede para salarios de € 500,00, € 1.000,00 y € 1,500,00.

En ❶ tenemos la variable `base` recibiendo una copia de `salario`. Esto es necesario porque, cuando atribuimos un nuevo valor a una variable, el valor anterior es sustituido (y perdido si no lo guardamos en otro lugar). Como vamos a utilizar el valor del salario digitado para exhibirlo en la pantalla, no podemos perderlo; por eso, la necesidad de una variable auxiliar llamada aquí `base`.

En ❷ verificamos si el valor de la variable `base` es mayor que € 3.000,00. Si esto es verdadero, ejecutamos las líneas ❸ y ❹. En ❸, calculamos 35% del valor superior a € 3.000,00. El resultado es almacenado en la variable `impuesto`. Como esa variable contiene el valor a pagar para esa cantidad, actualizaremos el valor de `base` para € 3.000,00 ❹, pues el que supera ese valor ya fue tarifado.

En ❺ verificamos si el valor de base es mayor que € 1.000,00, calculando 20% de impuesto en ❻, en caso que sea verdadero.

Veamos el rastreo para un salario de € 500,00:

salario	base	impuesto
500	500	0

Para un salario de € 1.500,00:

salario	base	impuesto
1500	1500	0
		100

Para un salario de € 3.000,00:

salario	base	impuesto
3000	3000	0
		400

Para un salario de € 5.000,00:

salario	base	impuesto
5000	5000	0
	3000	700
		1100

II. Consignas o preguntas reflexivas o actividades de resolución

II.1. Realizar el diagrama de flujo y el programa en Python para calcular el área de un rectángulo solamente cuando los lados tengan un valor positivo.

II.2. Realizar el diagrama de flujo y escriba un programa que pregunte el salario del empleado y calcule el valor del aumento. Para salarios superiores a € 1.250,00, calcule un aumento de 10%. Para los inferiores o iguales, de 15%.

Referencias bibliográficas consultadas y/o enlaces recomendados

- Countinho N (2008). Introducción a la programación con Python. 2da edición. Sao Paulo: Novatec Editora Ltda.

Caso 10:

Sentencia elif

Sección: ……..……..	Apellidos : ……..……
Docente :	Nombres : ……
Unidad: III	

Instrucciones: Leer el texto que se muestra a continuación y luego responder las preguntas del ítem II. Varios estudiantes elegidos al azar resuelven el problema en la pizarra.

I. Descripción o presentación del caso

A continuación, se presenta el análisis de la sentencia elif.

4.4 elif

Python presenta una solución muy interesante al problema de múltiples **if**s anidados. La cláusula **elif** sustituye un par **else if**, pero sin crear otro nivel de estructura, evitando problemas de desplazamientos innecesarios a la derecha.

Vamos a volver a considerar el problema de la lista 4.7, esta vez usando **elif**. Vea el resultado en el programa de la lista 4.8.

▶ **Lista 4.8 – Categoría x precio, usando elif**

```
categoría = int(input("Digite la categoría del producto:"))
if categoría == 1:
   precio = 10
elif categoría == 2:
   precio = 18
elif categoría == 3:
   precio = 23
elif categoría == 4:
   precio = 26
elif categoría == 5:
   precio = 31
else:
   print("¡Categoría inválida, digite un valor entre 1 y 5!")
   precio = 0
print("El precio del producto es: €%6.2f" % precio)
```

II. Consignas o preguntas reflexivas o actividades de resolución

II.1. Realizar el diagrama de flujo y el programa en Python para los ejercicios propuestos en la referencia.

Ejercicio 4.8 Escriba un programa que lea dos números y que pregunte qué operación desea realizar. Usted debe poder calcular la suma (+), sustracción (-), multiplicación (*) y división (/). Exhiba el resultado de la operación solicitada.

Ejercicio 4.9 Escriba un programa para aprobar el préstamo bancario para la compra de una casa. El programa debe preguntar el valor de la casa a comprar, el salario y la cantidad de años a pagar. El valor de la cuota mensual no puede ser superior a 30% del salario. Calcule el valor de la cuota dividiendo el valor de la casa a comprar por el número de meses a pagar.

Ejercicio 4.10 Escriba un programa que calcule el precio a pagar por el suministro de energía eléctrica. Pregunte la cantidad de kwh consumida y el tipo de instalación: R para residencias, I para industrias y C para comercios. Calcule el precio a pagar de acuerdo con la siguiente tabla.

Precio por tipo y rango de consumo		
Tipo	Faixa (kWh)	Precio
Residencial	hasta 500	€ 0,40
	más de 500	€ 0,65
Comercial	hasta 1000	€ 0,55
	más de 1000	€ 0,60
Industrial	hasta 5000	€ 0,55
	más de 5000	€ 0,60

Referencias bibliográficas consultadas y/o enlaces recomendados

- Countinho N (2008). Introducción a la programación con Python. 2da edición. Sao Paulo: Novatec Editora Ltda.

Caso 11:

Sentencia for y while

Sección: ……..……..	Apellidos : ……..……
Docente :	Nombres : ……
Unidad: III	

Instrucciones: Leer el texto que se muestra a continuación y luego responder las preguntas del ítem II. Varios estudiantes elegidos al azar resuelven el problema en la pizarra.

I. Descripción o presentación del caso

A continuación, se presenta el análisis de la sentencia for y while.

Sintaxis de la sentencia for

La sintaxis es la siguiente:

```
for iterador in secuenc
ia

    #codigo a
ejecutar
```

Esto quiere decir que cuando usamos la sentencia For, tenemos la capacidad de recorrer una secuencia por medio de "iteraciones", una secuencia como una lista o una simple cadena de texto, veamos un ejemplo para comprender mejor.

Si quisiéramos declarar una cadena de texto y recorrer cada uno de sus caracteres, podemos usar la sentencia For para ello.

Recorrer cadena de texto usando For

Este programa recorrerá cada letra de la cadena de texto "Hola!" y la imprimirá en pantalla.

```
#!/usr/bin/pyth
on

```

```
3 for letra in 'Hola!':

4     print 'Estamos en la letra :',
  letra
```

Este seria el resultado:

```
1 Estamos en la letra :
  H

2 Estamos en la letra :
  o



3 Estamos en la letra :
  l

4 Estamos en la letra :
  a



5 Estamos en la letra :
  !
```

Ciclo while en Python

En Python tenemos una palabra reservada llamada "**while**" que nos permite ejecutar ciclos, o bien secuencias periódicas que nos permiten ejecutar código múltiples veces.

El ciclo while nos permite realizar múltiples iteraciones basándonos en el resultado de una expresión lógica que puede tener como resultado un valor verdadero o falso (true o false).

Para utilizar este ciclo tenemos la siguiente sintaxis.

Sintaxis del ciclo while

```
1 #donde expresión es
  verdadero

2



3 while (expresión
  ):
```

```
4         #codigo..
.
```

En este ejemplo es importante comprender que "expresión" es una evaluación lógica que al dar como resultado un valor verdadero, se permitirá que el ciclo while siga ejecutándose continuamente, cuando la expresión cambie a un valor falso, el ciclo cesará. Veamos el siguiente ejemplo para comprenderlo mejor.

Ejemplo 1 – contador

```
1 #!/usr/bin/pyth
on

2



3 contador = 0

4 while (contador
< 5):



5    print 'El contador es :',
contador

6    contador = contador + 1



7

8 print "Fin del
programa"
```

En este ejemplo tenemos un contador con un valor inicial de cero, cada iteración del sitio while manipula esta variable de manera que incremente su valor en 1, por lo que después de su primera iteración el contador tendrá un valor de 1, luego 2, y así sucesivamente. Eventualmente cuando el contador llegue a tener un valor de 5, la condición del ciclo "contador < 5" será falsa, por lo que el ciclo terminará arrojando el siguiente resultado.

```
1 El contador es
: 0

2 El contador es
: 1



3 El contador es
: 2
```

```
El contador es
: 3


El contador es
: 4

Fin del programa
```

Interrupción de ciclos con break

Adicionalmente existe una forma alternativa de interrumpir o cortar los ciclos utilizando la palabra reservada "break". Esta nos permite salir del ciclo incluso si la expresión evaluada en while (o en otro ciclo como for) permanece siendo verdadera. Para comprender mejor usaremos el mismo ejemplo anterior pero interrumpiremos el ciclo usando break.

```
#!/usr/bin/pyth
on


contador = 0

while (contador
< 5):


    print 'El contador es :',
contador

    contador = contador + 1


    if (contador
> 3):

        break


print "Fin del
programa"
```

En este ejemplo evaluamos dentro del ciclo while si la condición de "contador > 3" es verdadera, al ser así se utilizará break para interrumpir el ciclo, de manera que el resultado de este programa sería el siguiente:

```
El contador es
: 0

El contador es
: 1


El contador es
: 2

Fin del programa
```

Espero que con este tutorial hayan podido comprender mejor la función while en python. No olviden dejar sus comentarios y recomendaciones, y seguir al tanto para más tutoriales.

II. Consignas o preguntas reflexivas o actividades de resolución

II.1. Realizar el diagrama de flujo y el programa en Python para calcular la factorial de un número.

II.2. Realizar el diagrama de flujo y el programa en Python para calcular la siguiente expresión.

$$□^{□} = 1 + \frac{□}{1!} + \frac{□^2}{2!} + \frac{□^3}{3!} + \cdots, \qquad -\infty < □ < \infty$$

II.3. Realizar el diagrama de flujo y el programa en Python para el enunciado de la pregunta 2.1.

Referencias bibliográficas consultadas y/o enlaces recomendados

- Códigoprogramación. Ciclo for en Python [en línea]. Disponible en http://codigoprogramacion.com/cursos/tutoriales-python/ciclo-for-en-python.html#.WoRs_yXOXIU, [Fecha de consulta: 01/02/2018].
- Códigoprogramación. Ciclo while en Python [en línea]. Disponible en http://codigoprogramacion.com/cursos/tutoriales-python/ciclo-while-en-python.html#.WoRu5yXOXIU, [Fecha de consulta: 01/02/2018].

Caso 12:

Funciones

Sección: ……..……..	Apellidos : ……..……
Docente :	Nombres : ……
Unidad: III	

Instrucciones: Leer el texto que se muestra a continuación y luego responder las preguntas del ítem II. Varios estudiantes elegidos al azar resuelven el problema en la pizarra.

I. Descripción o presentación del caso

A continuación, se presenta el análisis de una función en Python.

Definir funciones

Ahora que sabemos lo que se entiende por función, vamos a ver cómo podemos definir funciones en Python. Para definir una función, utilizaremos la declaración def, que tiene la siguiente sintaxis:

```
def function_name(parameters):
    statement(s)
    return expression
```

Los parámetros en la definición de una función son opcionales, ya que ciertas funciones no requieren parámetros que pasar en el momento de la llamada a la función. Si no se pasa más de un parámetro, los parámetros están separados por comas, y estarán vinculados a los argumentos de la función que se corresponden con los parámetros pasados. Las declaraciones (cuerpo de la función) se ejecutan cuando se llama a la función.

La instrucción de retorno es una declaración opcional que sirve valor de retorno cuando, al llamar a una función, se espera un valor devuelto. Si no se devuelve ningún valor específico, el valor devuelto por defecto será None.

Ejemplos

Vamos a ver algunos ejemplos para comprender aún más la idea de las funciones en Python. Te habrás dado cuenta de que al utilizar funciones estaremos evitando el repetirnos a nosotros mismos, ya que proporcionan un bloque de código reutilizable al cual llamar cada vez que queramos realizar alguna tarea periódica.

Vamos a ver un ejemplo muy sencillito para comprenderlo todo. En la función de a continuación, pasaremos el nombre del empleado, y la función lo mostrará por pantalla. El código sería tal que así:

```
employee_name = 'Abder'

def print_name(name):
    print name

print_name(employee_name)
```

Como puedes ver, al llamar a una función, podremos observar estos componentes:

- El nombre de la función (print_name)
- Los parámetros (employee_name)

Si escribes print_name(employee_name) antes de la definición de la función, Python se quejará de la siguiente manera:

```
Traceback (most recent call last):
  File "test.py", line 3, in < module >
    print_name(employee_name)
NameError: name 'print_name' is not defined
```

Por lo que, para que no ocurra esto, procura llamar a la función, siempre después de haberlo declarado.

II. Consignas o preguntas reflexivas o actividades de resolución

II.1. Realizar un programa en Python que utilice una función para calcular la factorial de un número.

II.2. Realizar un programa en Python que utilice una función para calcular el MCD de dos números.

Referencias bibliográficas consultadas y/o enlaces recomendados

- Programación.net. Funciones en Python [en línea]. Disponible en http://programacion.net/articulo/como_funcionan_las_funciones_en_python_1504, [Fecha de consulta: 01/02/2018].

Caso 13:

Funciones

Sección: ……..……..	Apellidos : ……..……
Docente :	Nombres : ……
Unidad: III	

Instrucciones: Leer el texto que se muestra a continuación y luego responder las preguntas del ítem II. Varios estudiantes elegidos al azar resuelven el problema en la pizarra.

I. Descripción o presentación del caso

A continuación, se presenta el análisis de una función en Python y el uso de una lista.

Definir funciones

Ahora que sabemos lo que se entiende por función, vamos a ver cómo podemos definir funciones en Python. Para definir una función, utilizaremos la declaración def, que tiene la siguiente sintaxis:

```
def function_name(parameters):
    statement(s)
    return expression
```

Los parámetros en la definición de una función son opcionales, ya que ciertas funciones no requieren parámetros que pasar en el momento de la llamada a la función. Si no se pasa más de un parámetro, los parámetros están separados por comas, y estarán vinculados a los argumentos de la función que se corresponden con los parámetros pasados. Las declaraciones (cuerpo de la función) se ejecutan cuando se llama a la función.

La instrucción de retorno es una declaración opcional que sirve valor de retorno cuando, al llamar a una función, se espera un valor devuelto. Si no se devuelve ningún valor específico, el valor devuelto por defecto será None.

Ejemplos

En este ejemplo utilizaremos listas. Asumiendo que tenemos la siguiente lista:

```
numbers_list = [1,2,3,4,5]
```

Esta función servirá para insertar nuevos números en la lista que pasaremos a la función:

```
numbers_list = [1,2,3,4,5]

def insert_numbers(numbers_list):
    numbers_list.insert(5, 8)
    numbers_list.insert(6, 13)
    print 'List "inside" the function is: ', numbers_list
    return

insert_numbers(numbers_list)
print 'List "outside" the function is: ', numbers_list
```

La salida de esta función sería:

```
List "inside" the function is:  [1, 2, 3, 4, 5, 8, 13]
List "outside" the function is:  [1, 2, 3, 4, 5, 8, 13]
```

¿Qué podemos extraer de todo esto? Podemos concluir que los parámetros se pasan por referencia. Es decir, los parámetros de la llamada a la función son los mismos que los argumentos pasados (variable/identidad) por parte de la llamada, en lugar de pasarlos por valor, donde los llamados parámetros de la función son una copia de llamada.

II. Consignas o preguntas reflexivas o actividades de resolución

II.1. Realizar un programa en Python que utilice una función para calcular el promedio de los elementos de una lista.

Referencias bibliográficas consultadas y/o enlaces recomendados

- Programación.net. Funciones en Python [en línea]. Disponible en http://programacion.net/articulo/como_funcionan_las_funciones_en_python_1504, [Fecha de consulta: 01/02/2018].

Caso 14:

Desarrollo de prototipos

Sección: ……..……..	Apellidos : ……..……
Docente :	Nombres : ……
Unidad: III	

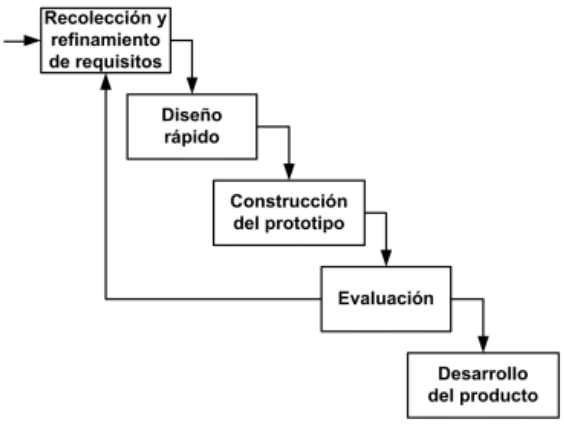
Instrucciones: Leer el texto que se muestra a continuación y luego responder las preguntas del ítem II. Varios estudiantes elegidos al azar resuelven el problema en la pizarra.

I. Descripción o presentación del caso

A continuación, se presenta el análisis del desarrollo de prototipos.

Desarrollo de prototipos

Normalmente, el cliente es capaz de definir un conjunto general de objetivos para el sistema que hemos de construir, pero no identifica los requisitos detallados. En otros casos, puede que nosotros no estemos seguros de la eficiencia de un algoritmo, de la capacidad de nuestro diseño para soportar los requerimientos del sistema o de la forma en que debe diseñarse la interfaz de usuario. En cualquiera de estas situaciones, resulta adecuado construir un prototipo.



Prototipado

El desarrollo de prototipos reduce el riesgo de que nuestro proyecto fracase y facilita la especificación de requerimientos de productos que desconocemos. Sin embargo, también tiene sus inconvenientes: el cliente puede pensar que el prototipo es el sistema definitivo, ignorando que un prototipo no es un sistema acabado aunque tenga el mismo aspecto externo. Esto puede conducir a la consolidación de aspectos de baja calidad de un prototipo en el

sistema final que se entrega si el prototipo no se desecha a tiempo.

Fred Brooks nos aclara lo que hay que hacer cuando un prototipo ya ha cumplido con su propósito: *"En la mayoría de los proyectos, el primer sistema que se construye apenas resulta utilizable. Puede que sea demasiado lento, demasiado grande, difícil de usar o las tres cosas a la vez. No queda más remedio que comenzar de nuevo y construir una versión rediseñada que resuelva los problemas... Cuando se utiliza un concepto nuevo... hay que construir un sistema para desecharlo, porque incluso la mejor planificación no puede asegurar que vaya a salir bien la primera vez. Por tanto, la cuestión no es si hay que construir un sistema piloto y desecharlo. Se desechará. La única cuestión es si planificar de antemano la construcción de algo que se va a desechar, o prometer la entrega del desecho a los clientes..."* (*The Mythical Man-Month*, "El mítico hombre-mes", 1975, uno de los libros de gestión de proyectos de desarrollo de software más populares que jamás se han escrito).

A veces, los prototipos desechables no se llegan a desechar. Pero los prototipos no siempre son desechables. En tal caso, estaremos utilizando un modelo iterativo de refinamiento de prototipos en el que, tras varias iteraciones, seremos capaces de construir un sistema que se adapte mejor a las necesidades de nuestro cliente.

II. Consignas o preguntas reflexivas o actividades de resolución

II.1. ¿Cómo aplicaría el ciclo de vida de prototipo para un programa que realice las funciones de una calculadora básica?

Referencias bibliográficas consultadas y/o enlaces recomendados

- Berzal F. El ciclo de vida de un sistema de información [en línea]. Disponible en http://flanagan.ugr.es/docencia/2005-2006/2/apuntes/ciclovida.pdf, [Fecha de consulta: 01/02/2018].