GLOSARIO BÁSICO

Contenido

Sistema Operativo 3
Archivos 3
Extensiones de archivos 3
Carpetas / directorios 3
Unidades de información y conversiones 4
Estructuras Discretas 4
Operadores 4
Expresiones aritméticas y lógicas 6
Variables e Identificadores 7
Parámetros / Argumentos 8
Constantes 8
Tipos de datos 8
Estructuras de Control 8
Programación 9
Procesos universales 9
Lenguaje de Programación 10
Algoritmos en lenguaje natural, pseudolenguaje y lenguaje formal 10
Sintaxis 10
Programa compilado 10
Programa interpretado 11
Depuración de programas 11
Momentos/tiempos de la programación: edición, compilación, interpretación, ejecución/corrida 11
Corridas en frío 11
Errores: de compilación / lógicos 12
Programación Orientada a Objetos 12
Aplicaciones 12
Cuentas y sesiones 12
IDEs 12
GIT 13
APIs / Extensiones / Librerías 13

Repositorio 13
Conceptos 14
Análisis 14
Metodología 14
Modelos / Prototipos 14
Paradigma 14

SISTEMA OPERATIVO

Un sistema operativo (SO) es el conjunto de programas de un sistema informático que gestiona los recursos del hardware y provee servicios a los programas de aplicación de software.

Sistemas operativos de interés para los estudiantes de la UCLA:

- ✓ Windows
- ✓ Linux
- ✓ Android
- ✓ IOs

Fuente: https://es.wikipedia.org/wiki/Sistema_operativo

ARCHIVOS

Un archivo (o fichero) informático es una secuencia de bytes almacenados en un dispositivo. Un archivo es identificado por un nombre y la descripción de la carpeta o directorio que lo contiene. A los archivos informáticos se les llama así porque son los equivalentes digitales de los archivos escritos en expedientes, tarjetas, libretas, papel o microfichas del entorno de oficina tradicional.

Un archivo de datos informático normalmente tiene un tamaño, que generalmente se expresa en bytes. Depende del software que se ejecuta en la computadora el interpretar esta estructura básica como por ejemplo un programa, un texto o una imagen, basándose en su nombre, extensión y contenido.

Fuente: https://es.wikipedia.org/wiki/Archivo_(informática)

EXTENSIONES DE ARCHIVOS

En informática, el término extensión del fichero (también, extensión del archivo) es una cadena de caracteres anexada al nombre de un archivo, habitualmente predicha por un punto. Su función principal es distinguir el contenido del archivo, de modo que el sistema operativo disponga del procedimiento necesario para ejecutarlo o interpretarlo.

Sin embargo, la extensión es solamente parte del nombre del archivo y no representa ningún tipo de obligación respecto a su contenido.

Fuente: https://es.wikipedia.org/wiki/Extensión_de_archivo

CARPETAS / DIRECTORIOS

En informática, un directorio o carpeta2 es un contenedor virtual en el que se almacenan una agrupación de archivos informáticos y otros subdirectorios, atendiendo a su contenido, a su propósito o a cualquier criterio que decida el usuario.

En el entorno gráfico de los sistemas operativos modernos, el directorio se denomina metafóricamente carpeta y de hecho se representa con un icono con esta figura. Esta imagen se asocia con el ambiente administrativo de cualquier oficina, donde la carpeta de cartón encierra las hojas de papel (representando a los archivos de datos) de un expediente.

Fuente: https://es.wikipedia.org/wiki/Directorio

UNIDADES DE INFORMACIÓN Y CONVERSIONES

En las matemáticas puras, un valor no tiene un límite de espacio para su representación, sin embargo, las computadoras generalmente trabajan con un número fijo de bits.

A la unidad mínima de información en una computadora se le llama bit. Con un bit se puede representar sólo un valor de dos posibles valores diferentes; por ejemplo: cero o uno, falso o verdadero, blanco o negro, abajo o arriba, no o sí, etc.

Un byte es una unidad de información formada por una secuencia de bits adyacentes.

Múltiplos de bytes			
Sistema Internacional (decimal)		ISO/IEC 80000-13 (binario)	
Múltiplo (símbolo)	SI	Múltiplo (símbolo)	ISO/IEC
kilobyte (kB)	10^3	*kibibyte* (KiB)	2^{10}
megabyte (MB)	10^6	*mebibyte* (MiB)	2^{20}
gigabyte (GB)	10^9	*gibibyte* (GiB)	2^{30}
terabyte (TB)	10^{12}	*tebibyte* (TiB)	2^{40}
petabyte (PB)	10^{15}	*pebibyte* (PiB)	2^{50}
exabyte (EB)	10^{18}	*exbibyte* (EiB)	2^{60}
zettabyte (ZB)	10^{21}	*zebibyte* (ZiB)	2^{70}
yottabyte (YB)	10^{24}	*yobibyte* (YiB)	2^{80}

Fuente: https://es.wikipedia.org/wiki/Byte

ESTRUCTURAS DISCRETAS

OPERADORES

Un operador es un símbolo matemático que indica que debe ser llevada a cabo una operación especificada1 sobre un cierto número de operandos (número, función, vector, etc.).

Los operadores suelen interpretarse como funciones, mejor aún como aplicaciones, por ejemplo la adición, la multiplicación, etc., pueden ser entendidas como funciones de dos argumentos.

- Operadores de condición

Relacionan un término A con otro B estableciendo su igualdad, jerarquía o cualquier otra relación posible, como ejemplos tenemos:

- ✓ A = B establece que A es igual que B. Hay que distinguir entre operador de asignación y operador de comparación. En el lenguaje JavaScript A = B significa que debe tomarse el valor de B y asignárselo a A; para establecer una comparación se usaría A == B o A === B (doble o triple =)
- ✓ A ≠ B o desigualdad: el resultado de esta operación será FALSO si los valores A y B son iguales y VERDADERO si son distintos. En JavaScript se denota así: A !== B

- Operadores de orden

Los operadores de orden establecen o verifican clasificaciones entre números (A < B, A > B, etc.) u otro tipo de valores (caracteres, cadenas, ...)

Todo tipo de dato susceptible de ser ordenado por cualquier criterio puede ser comparado con estos operadores; como los anteriores devuelven un valor de verdad en función del resultado que tenga la comparación en cada caso.

- ✓ A > B Devuelve **VERDADERO** si **A** es estrictamente mayor que **B** y **FALSO** en caso contrario
- ✓ A < B Devuelve **VERDADERO** si **A** es estrictamente menor que **B** y **FALSO** en caso contrario
- ✓ A ≥ B Devuelve **VERDADERO** si **A** es mayor o igual que **B** y **FALSO** en caso contrario
- ✓ A ≤ B Devuelve **VERDADERO** si **A** es menor o igual que **B** y **FALSO** en caso contrario

- Operadores lógicos

Muy utilizados en Informática, lógica proposicional y álgebra booleana, entre otras disciplinas. Nos proporcionan un resultado a partir de que se cumpla o no una cierta condición. Esto genera una serie de valores que, en los casos más sencillos, pueden ser parametrizados con los valores numéricos 0 y 1, como se puede apreciar en los ejemplos de abajo. La combinación de dos o más operadores lógicos conforma una función lógica.

Los más sencillos son (nótese su relación con los operadores relacionales):

- ✓ **Operador NO-lógico**: '¬A' significa todo lo que no es A'. **En JavaScript: !A**
- ✓ **Operador Y-lógico**: 'A ∧ B' significa 'A y B a la vez'; resultando FALSO (0) si no se cumple y VERDADERO (1) si lo hace. **En JavaScript: A && B**
- ✓ **Operador O-lógico**: 'A o B' si A es verdadera y B es verdadera, el resultado es verdadero, si A es falso y B es verdadero o viceversa sigue siendo verdadero y si A y B son falsos da igual a falso significa; 'O bien A, o bien B, o bien los dos'; resultando FALSO (0) si no se dan ni A ni B y VERDADERO (1) si se da alguno de los dos o los dos a la vez. **En JavaScript: A || B**
- ✓ **Operador =**: 'A = B' significa 'A debe ser igual a B'; resultando FALSO (0) si esto no es así y VERDADERO (1) en caso contrario. **En JavaScript: A == B o A === B**
- ✓ **Operador <**: 'A < B' significa 'A debe ser menor que B'; resultando FALSO (0) si no se satisface y VERDADERO (1) en caso contrario. **En JavaScript: A < B**
- ✓ **Operador >**: 'A > B' significa 'A debe ser mayor que B'; resultando FALSO (0) si no se satisface y VERDADERO (1) en caso contrario. **En JavaScript: A > B**

Los operadores más complejos se construyen a partir de los anteriores (podría incluirse alguno más) y ya entran dentro de lo que sería una función lógica. Un ejemplo muy utilizado sería 'SI(condición;A;B)' ('IF condición THEN A ELSE B' en la mayoría de los lenguajes de programación) cuyo resultado sería A si se satisface la 'condición' o B en caso contrario.

Fuente: https://es.wikipedia.org/wiki/Operador

EXPRESIONES ARITMÉTICAS Y LÓGICAS

En informática y lenguajes de programación, se entiende por expresión aritmética a aquella donde los operadores que intervienen en ella son numéricos, el resultado es un número y los operadores son aritméticos.

Los operadores aritméticos más comúnmente utilizados son:

- ✓ El signo más (+) se emplea para sumar dos valores
- ✓ El signo menos (-) para restar un valor de otro
- ✓ El asterisco (*) para multiplicar dos valores
- ✓ La división (/) para dividir un valor por otro
- ✓ El signo % para obtener el resto de una división entera.

Estos símbolos se conocen como operadores binarios, pues operan sobre dos valores o variables.

La lista siguiente son ejemplos de expresiones aritméticas:

- ✓ resultado = x - y;
- ✓ total = capital+ interés;
- ✓ cuadrado = x * x;
- ✓ celcius = (fahrenheit - 32) / 1.8

Hay que comprender que el signo igual (=) en las expresiones anteriores se le conoce como "operador de asignación". Asigna el valor de la derecha de dicho signo igual a la variable de la izquierda.

En la última expresión, se utilizan paréntesis () para realizar primero cierta operación. Esto sucede porque en lenguajes de programación, los operadores siguen unas reglas de preferencia. *, / y % tienen preferencia sobre + y -. Para soslayar esta preferencia, se deben utilizar paréntesis.

Las expresiones con operadores de la misma preferencia se suelen evaluar de izquierda a derecha.

Otro punto a tener en cuenta es que en una expresión que entraña una división, hay que tener cuidado de evitar la división por cero, que da como resultado infinito o un valor anómalo.

Fuente: https://es.wikipedia.org/wiki/Expresión_aritmética

En programación, una expresión es una combinación de constantes, variables o funciones, que es interpretada de acuerdo a las normas particulares de precedencia y asociación para un lenguaje de programación en particular. Como en matemáticas, la expresión es su valor evaluado, es decir, la expresión es una representación de ese valor.

Ejemplos de expresiones

- ✓ Expresión relacional: y>8
- ✓ Expresión aritmética: 3+2, x+1,...

✓ Expresión lógica: x OR y, NOT x,...

Fuente: https://es.wikipedia.org/wiki/Expresión_(informática)

Las expresiones lógicas en programación son aquellas que pueden tomar uno de dos valores, verdadero o falso Estas expresiones se utilizan en las estructuras de control de flujo para tomar decisiones basadas en los valores booleanos. Las expresiones lógicas se forman combinando constantes y variables con operadores lógicos y relacionales.

Los operadores lógicos más comunes son:

- ✓ AND (&&): Devuelve verdadero si ambos valores son verdaderos, de lo contrario devuelve falso
- ✓ OR (||): Devuelve verdadero si al menos uno de los valores es verdadero, de lo contrario devuelve falso.
- ✓ NOT (!): Devuelve el valor opuesto al valor booleano dado, es decir, si el valor es verdadero, devuelve falso, y si el valor es falso, devuelve verdadero

Los operadores relacionales se utilizan para expresar condiciones y comparar los valores de dos expresiones.

- ✓ Mayor que (>): Devuelve verdadero si la expresión de la izquierda es mayor que la expresión de la derecha
- ✓ Menor que (<): Devuelve verdadero si la expresión de la izquierda es menor que la expresión de la derecha.
- ✓ Igual que (==): Devuelve verdadero si la expresión de la izquierda es igual a la expresión de la derecha
- ✓ Diferente que (!=): Devuelve verdadero si la expresión de la izquierda es diferente a la expresión de la derecha

Operador	Significado
<	menor que
>	mayor que
=, ==	igual que
<=	menor o igual que
>=	mayor o igual que
<>, !=	distinto de

Fuente: https://issuu.com/denissecedenom05gmail.com/docs/revista-_programaci_n/s/25323154

VARIABLES E IDENTIFICADORES

En programación, una variable está formada por un espacio en el sistema de almacenaje (memoria principal de un ordenador) y un nombre simbólico (un identificador) que está asociado a dicho espacio. Ese espacio contiene una cantidad de información conocida o desconocida, es decir un valor.

El nombre de la variable es la forma usual de referirse al valor almacenado: esta separación entre nombre y contenido permite que el nombre sea usado independientemente de la información exacta que representa.

El identificador, en el código fuente de la computadora puede estar ligado a un valor durante el tiempo de ejecución y el valor de la variable puede por lo tanto cambiar durante el curso de la ejecución del programa.

Fuente: https://es.wikipedia.org/wiki/Variable_(programación)

PARÁMETROS / ARGUMENTOS

En ciencias de la computación, un parámetro es una variable utilizada para recibir valores de entrada en una rutina, subrutina o método. Dichos valores, que serán enviados desde la rutina invocante, son llamados argumentos.

La subrutina usa los valores asignados a sus parámetros para alterar su comportamiento en tiempo de ejecución. La mayor parte de los lenguajes de programación pueden definir subrutinas que aceptan cero o más argumentos.

Fuente: https://es.wikipedia.org/wiki/Argumento_(informática)

CONSTANTES

En programación, una constante es un valor que no puede ser alterado/modificado durante la ejecución de un programa, únicamente puede ser leído.

Una constante corresponde a una longitud fija de un área reservada en la memoria principal del ordenador, donde el programa almacena valores fijos.

Por ejemplo:

- ✓ El valor de PI = 3,1416

Por conveniencia, el nombre de las constantes suele escribirse en mayúsculas en la mayoría de lenguajes.

Fuente: https://es.wikipedia.org/wiki/Constante_(informática)

TIPOS DE DATOS

Debido a que las variables contienen o apuntan y muestran a valores de tipos determinados, las operaciones sobre las mismas y el dominio de sus propios valores están determinadas por el tipo de datos en cuestión. Algunos tipos de datos usados:

- ✓ Tipo de **dato lógico** (verdadero/1 o falso/0).
- ✓ Tipo de **dato entero** (real, sin decimales).
- ✓ Tipo de **dato de coma flotante** (real, con decimales).
- ✓ Tipo de **dato carácter** (letra o dígito).
- ✓ Tipo de **dato cadena** (cadena de texto(s)).

ESTRUCTURAS DE CONTROL

En lenguajes de programación, las estructuras de control permiten modificar el flujo de ejecución de las instrucciones de un programa.

Con las estructuras de control se puede:

- ✓ De acuerdo con una condición, ejecutar un grupo u otro de sentencias (If-Then-Else)
- ✓ De acuerdo con el valor de una variable, ejecutar un grupo u otro de sentencias (Switch-Case)
- ✓ Ejecutar un grupo de sentencias solo cuando se cumpla una condición (Do-While)
- ✓ Ejecutar un grupo de sentencias hasta que se cumpla una condición (Do-Until)
- ✓ Ejecutar un grupo de sentencias un número determinado de veces (For-Next)

Todas las estructuras de control tienen un único punto de entrada. Las estructuras de control se pueden clasificar en: secuenciales, iterativas y de control avanzadas. Esta es una de las cosas que permiten que la programación se rija por los principios de la programación estructurada.

Los lenguajes de programación modernos tienen estructuras de control similares. Básicamente lo que varía entre las estructuras de control de los diferentes lenguajes es su sintaxis; cada lenguaje tiene una sintaxis propia para expresar la estructura.

Fuente: https://es.wikipedia.org/wiki/Estructuras_de_control

PROGRAMACIÓN

PROCESOS UNIVERSALES

En programación, hay una serie de procesos de uso muy común en todo ámbito de la Informática. Se trata de los pasos necesarios para el cálculo de los siguientes indicadores:

- ✓ **Contadores**: es el valor que indica la cantidad de veces que ocurre algo, bien sea en relación al total de los elementos siendo evaluados, o la cantidad de esos elementos que cumplen con alguna condición. El proceso es: **iniciando el contador en cero, por cada elemento sumar 1 al contador**.
- ✓ **Acumuladores**: es una versión ampliada de los contadores, pero por cada elemento el valor de incremento es variable. Ej.: la cantidad total de dinero que tiene un grupo de personas. El proceso es: **iniciando el acumulador en cero, por cada elemento sumar un valor al acumulador** (del ejemplo: la cantidad de dinero).
- ✓ **Mayor / Menor**: consiste en determinar el mayor / menor valor entre varios elementos siendo evaluados. El valor mayor se inicia en un número pequeño, y el menor en un número grande. El proceso es: **por cada elemento si el valor es ahora más / menos grande, entonces cambiar el valor mayor / menor**.
- ✓ **Dato relativo al Mayor / Menor**: es similar al proceso de mayor / menor, pero ahora queremos determinar un dato asociado al elemento que tiene el mayor. Ej.: el nombre de la persona con mayor edad. El proceso sólo debe ajustarse de manera que **al cambiar el mayor / menor, entonces también cambia el dato relativo**.
- ✓ **Promedio**: lo obtenemos al sumar todos los datos y dividir entre la cantidad de datos. El proceso: **por cada elemento, acumular el valor y contar el elemento**. La fórmula del promedio es: **acumulador de valores / contador de valores**.
- ✓ **Porcentaje**: es una forma de cuantificar un pedazo de una cierta cantidad y nos representa una fracción de un todo de 100 partes iguales. Ej.: entre un grupo de 20 personas, si 12 son mujeres entonces el 60% de las personas son mujeres. Para el cálculo de porcentaje intervienen un valor parcial y un valor total. El proceso: **por cada elemento contar en el valor parcial si cumple la condición que se pide, el valor total se incrementa para cada elemento del todo**. La fórmula del

porcentaje es: **valor parcial / valor total * 100**, del ejemplo: 12 mujeres / 20 personas * 100 = 60%. En este caso el valor parcial son las mujeres, el valor total son todas las personas.

Fuente: Guía Estructuras preliminares, del DCyT - UCLA

LENGUAJE DE PROGRAMACIÓN

Es un lenguaje formal (o artificial, es decir, un lenguaje con reglas gramaticales bien definidas) que proporciona a una persona, en este caso el programador, la capacidad y habilidad de escribir (o programar) una serie de instrucciones o secuencias de órdenes en forma de algoritmos con el fin de controlar el comportamiento físico o lógico de un sistema informático, para que de esa manera se puedan obtener diversas clases de datos o ejecutar determinadas tareas.

A todo este conjunto de órdenes escritas mediante un lenguaje de programación se le denomina programa informático.

Fuente: https://es.wikipedia.org/wiki/Lenguaje_de_programación

ALGORITMOS EN LENGUAJE NATURAL, PSEUDOLENGUAJE Y LENGUAJE FORMAL

Podemos expresar un algoritmo de muchas maneras, incluyendo lenguaje natural, diagramas de flujo, pseudocódigo y, por supuesto, en lenguajes de programación reales.

- **Lenguaje natural**: es decir nuestro lenguaje hablado y escrito. Se nos da naturalmente y puede comunicar los pasos de un algoritmo a una audiencia general. Cuando desarrollamos algoritmos, a menudo trabajamos con personas que saben programación y con algunos que no; pero todos conocen el lenguaje natural.
- **Pseudolenguaje**: es un código que utiliza construcciones de un lenguaje de programación, pero que en realidad, no se ejecuta.
- **Lenguaje formal**: Después de planear nuestro algoritmo, ya sea en lenguaje natural, diagramas de flujo, pseudocódigo, o una combinación de esos, es momento de convertirlo en código ejecutable, como JavaScript.

Fuente: https://es.khanacademy.org/computing/ap-computer-science-principles

SINTAXIS

La sintaxis de un lenguaje informático es el conjunto de reglas que define las combinaciones de símbolos que se consideran declaraciones o expresiones correctamente estructuradas en ese lenguaje. Esto se aplica tanto a los lenguajes de programación, donde el documento representa el código fuente, como a los lenguajes de marcado, donde el documento representa los datos.

Fuente: https://es.wikipedia.org/wiki/Sintaxis_(lenguajes_de_programación)

PROGRAMA COMPILADO

Un lenguaje compilado es un lenguaje de programación cuya implementación es normalmente con el uso de compiladores (traductores que generan código de máquina a partir del código fuente).

Es decir, el programa compilado queda a nivel del código de la máquina (ceros y unos), y se obtiene gracias a que el compilador convierte el código fuente (ej. Programa en C++) al código máquina.

Fuente: https://es.wikipedia.org/wiki/Lenguaje_de_programación_compilado

PROGRAMA INTERPRETADO

En ciencias de la computación, intérprete o interpretador es un programa informático capaz de analizar y ejecutar otros programas. Los intérpretes se diferencian de los compiladores o de los ensambladores en que mientras estos traducen un programa desde su descripción en un lenguaje de programación al código de máquina del sistema, los intérpretes solo realizan la traducción a medida que sea necesaria, típicamente, instrucción por instrucción, y normalmente no guardan el resultado de dicha traducción.

Fuente: https://es.wikipedia.org/wiki/Intérprete_(informática)

DEPURACIÓN DE PROGRAMAS

La depuración de programas es el proceso de identificar y corregir errores de programación.1 Es conocido también por el término inglés «debugging», cuyo significado es eliminación de «bugs», esta es la manera en que se conoce informalmente a los errores de programación.

Fuente: https://es.wikipedia.org/wiki/Depuración_de_programas

MOMENTOS/TIEMPOS DE LA PROGRAMACIÓN: EDICIÓN, COMPILACIÓN, INTERPRETACIÓN, EJECUCIÓN/CORRIDA

Los momentos en la programación tienen que ver con cual etapa de la creación de un programa nos referimos al hablar del proyecto que llevamos a cabo.

- ✓ **Tiempo de edición**: nos referimos al programa que estamos construyendo en el editor.
- ✓ **Tiempo de compilación / interpretación**: se trata de las revisiones que hace el lenguaje de programación del programa que hacemos, y nos indica los errores sintácticos que posee el programa. Una vez que el lenguaje nos dice que no hay errores, entonces tenemos un programa sintácticamente correcto.
- ✓ **Tiempo de ejecución / corrida**: cuando corremos nuestro programa, para introducir los datos y obtener los resultados, los mismos pueden estar errados dado que la lógica que aplicamos estaría equivocada. Entonces, los errores en los resultados se producen cuando ejecutamos el programa, en tiempo de ejecución.

Gracias a los editores modernos, el tiempo de edición y de compilación prácticamente se fusionan, de manera que los errores se van señalando en la medida que escribimos, y automáticamente desaparecen cuando los corregimos.

Fuente: https://platzi.com/discusiones/1543-kotlin

CORRIDAS EN FRÍO

Es la ejecución manual de cada sentencia del programa, utilizando un conjunto de datos de entrada determinados y verificando que los resultados son correctos. Como una técnica de depuración, el programador debe utilizar datos que permitan recorrer todas las posibles rutas del programa.

Fuente: http://webdelprofesor.ula.ve/ingenieria/gilberto/pr1/04_EstructuraDelPrograma.pdf

ERRORES: DE COMPILACIÓN / LÓGICOS

- **Errores de sintaxis (compilación)**: Son todos aquellos que se generan por infringir las normas de escritura de un lenguaje. Suelen deberse a olvidos o a desconocimiento (programadores principiantes) y comprenden falta o mal uso de elementos separadores (comas, puntos y comas, dos puntos, etc.), palabras mal escritas (por ejemplo Mietras en vez de Mientras o Finelizar en vez de Finalizar).
- **Errores lógicos**: Son todos aquellos derivados de un mal diseño de los algoritmos o la arquitectura modular. Las posibilidades son muchas y los efectos posibles también. Pueden ir desde el bloqueo o detención indeseada del programa hasta un resultado incorrecto, una parte del programa que no se ejecuta, etc.

Fuente: https://www.aprenderaprogramar.com/index.php

PROGRAMACIÓN ORIENTADA A OBJETOS

La programación orientada a objetos (POO) es un paradigma de programación que parte del concepto de "objetos" como base, los cuales contienen información en forma de campos (a veces también referidos como atributos cualidades o propiedades) y código en forma de métodos.

Fuente: https://es.wikipedia.org/wiki/Programación_orientada_a_objetos

APLICACIONES

CUENTAS Y SESIONES

Una sesión es una interacción entre un sitio o una aplicación y un visitante que ha cargado por lo menos una página. Si no se realiza ninguna acción dentro de un período de tiempo definido (30 minutos de forma predeterminada), la visita termina. A diferencia de las visitas, las sesiones incluyen todos los eventos registrados desde el inicio de una visita.

Las cuentas de usuario son aquellas que le dan a una persona acceso al equipo y a los programas que este contiene. Para ingresar es necesario un nombre para la cuenta y una contraseña. Las cuentas de usuario son creadas por el administrador del equipo, quien se encargará de darle el acceso a otras personas.

Fuente: https://edu.gcfglobal.org/es/como-usar-windows-10

IDEs

Es una aplicación informática que proporciona servicios integrales para facilitar al desarrollador o programador el desarrollo de software.

Normalmente, un IDE consiste en un editor de código fuente, herramientas de construcción automáticas y un depurador. La mayoría de los IDE tienen auto-completado inteligente de código (IntelliSense).

Las IDEs modernas incluyen Inteligencia Artificial.

Fuente: https://es.wikipedia.org/wiki/Entorno_de_desarrollo_integrado

GIT

Git es un software de control de versiones pensando en la eficiencia, la confiabilidad y compatibilidad del mantenimiento de versiones de aplicaciones cuando estas tienen un gran número de archivos de código fuente.

Su propósito es llevar registro de los cambios en archivos de computadora incluyendo coordinar el trabajo que varias personas realizan sobre archivos compartidos en un repositorio de código.

Fuente: https://es.wikipedia.org/wiki/Git

APIS / EXTENSIONES / LIBRERÍAS

Las API son mecanismos que permiten a dos componentes de software comunicarse entre sí mediante un conjunto de definiciones y protocolos.

Por ejemplo, el sistema de software del instituto de meteorología contiene datos meteorológicos diarios. La aplicación meteorológica de su teléfono "habla" con este sistema a través de las API y le muestra las actualizaciones meteorológicas diarias en su teléfono.

Fuente: https://aws.amazon.com/es/what-is/api

Las extensiones son aplicaciones que se instalan en otras aplicaciones y que cumplen funciones específicas requeridas por el usuario. Las extensiones pueden mejorar las prestaciones de las aplicaciones que las usan.

Fuente: https://ayudica.unizar.es/otrs/public.pl

En informática, una biblioteca o, llamada por vicio del lenguaje, librería (del inglés library) es un conjunto de implementaciones funcionales, codificadas en un lenguaje de programación, que ofrece una interfaz bien definida para la funcionalidad que se invoca.

A diferencia de un programa ejecutable, el comportamiento que implementa una biblioteca no espera ser utilizada de forma autónoma (un programa sí: tiene un punto de entrada principal), sino que su fin es ser utilizada por otros programas, independientes y de forma simultánea.

Fuente: https://es.wikipedia.org/wiki/Biblioteca_(informática)

REPOSITORIO

Un repositorio, o repo, es un tipo de almacenamiento digital centralizado que los desarrolladores utilizan para realizar y administrar cambios en el código fuente de una aplicación. Los desarrolladores tienen que almacenar y compartir carpetas, archivos de texto y otros tipos de documentos al desarrollar software. Un repositorio cuenta con características que permiten a los desarrolladores rastrear con facilidad cambios en el código, editar archivos de manera simultánea y colaborar de forma eficiente en el mismo proyecto desde cualquier ubicación.

Fuente: https://aws.amazon.com/es/what-is/repo

CONCEPTOS

ANÁLISIS

Un programador es una persona que resuelve problemas, y para llegar a ser un programador eficaz se necesita aprender a resolver problemas de un modo riguroso y sistemático:

- ✓ **Análisis del problema**: consiste en el estudio detallado del problema. Se debe identificar los datos de entrada, de salida y la descripción del problema.
- ✓ **Diseño del algoritmo**: que describe la secuencia ordenada de pasos que conduce a la solución de un problema dado: algoritmo.
- ✓ **Transformación del algoritmo en un programa (codificación)**: Se expresa el algoritmo como un programa en un lenguaje de programación.
- ✓ **Ejecución y validación del programa**.

Fuente: https://plataforma.josedomingo.org/pledin/cursos/curso_cpp1/curso/u01/

METODOLOGÍA

La metodología de desarrollo de software es el conjunto de técnicas y métodos que se utilizan para diseñar una solución de software informático. Es importante señalar que existen varias, de manera que es una decisión de cada equipo.

Trabajar con una metodología es imprescindible por una cuestión de organización. No en vano, los factores tienen que estar ordenados y saber cómo se van a utilizar.

Por otra parte, las metodologías también sirven para controlar el desarrollo del trabajo. Esto sirve para minimizar los márgenes de errores y anticiparse a esa situación.

Fuente: https://www.universitatcarlemany.com/actualidad

MODELOS / PROTOTIPOS

Un modelo prototipo o modelo de desarrollo evolutivo es utilizado principalmente en el desarrollo de software para ofrecer al usuario una visión previa de cómo será el programa o sistema. Se le dice de desarrollo evolutivo al modelo de prototipo porque evoluciona hasta convertirse en el producto final.

Fuente: https://www.hostingplus.pe/blog/modelo-de-prototipos

PARADIGMA

Un paradigma de programación es una manera o estilo de programación de software. Existen diferentes formas de diseñar un lenguaje de programación y varios modos de trabajar para obtener los resultados que necesitan los programadores. Se trata de un conjunto de métodos sistemáticos aplicables en todos los niveles del diseño de programas para resolver problemas computacionales.

Los lenguajes de programación adoptan uno o varios paradigmas en función del tipo de órdenes que permiten implementar como, por ejemplo, Python o JavaScript, que son multiparadigmas.

Fuente: https://profile.es/blog/que-son-los-paradigmas-de-programacion/