



INSTITUTO TECNOLÓGICO SUPERIOR DE SAN ANDRÉS TUXTLA



FUNDAMENTOS DE LA PROGRAMACION ORIENTADA A OBJETOS

PLAN DE ESTUDIO: INGENIERIA INFORMATICA

ALUMNO: EMMANUEL DE JESUS TEOBAL DIAZ

DOCENTE: ENRIQUE TELONA TORRES

18/09/2022

CONTENIDO

INTRODUCCION-----	4
1.1 EVOLUCION DE LA PROGRAMACION -----	5
1.1.1 ¿QUE ES LA PROGRAMACION? -----	5
1.1.2 EL PRIMER PROGRAMADOR DE LA HISTORIA -----	6
1.1.3 LA TARJETA PERFORADA -----	6
1.1.4 PROGRAMACION MAS CERCANA A LA ACTUALIDAD-----	7
1.2 CONCEPTOS FUNDAMENTALES DE LA PROGRAMACION ORIENTADA A OBJETOS-----	8
1.2.1 PRINCIPIO DE ENCAPSULACION-----	8
1.2.2 PRINCIPIO DE ABSTRACCION-----	9
1.2.3 PRINCIPIO DE HERENCIA-----	9
1.2.4 PRINCIPIO DE POLIMORFISMO-----	9
1.3 LENGUAJE ORIENTADO A OBJETOS-----	11
1.4 RELACION ENTRE CLASES Y OBJETOS-----	12
1.5 PAPEL DE CLASE Y OBJETO EN EL ANALISIS Y DISEÑO-----	13
1.5.1 ELEMENTOS DEL MODELO DE OBJETOS-----	13
1.5.1.2 MODULARIDAD-----	13
1.5.1.3 JERARQUIA-----	13
1.5.1.4 TIPIFICACION-----	14
1.5.1.5 CONCURRENCIA-----	14

1.6 ENTORNOS DE PROGRAMACION-----	15
1.6.1 EDITORES DE TEXTO-----	15
1.6.2 PROCESADORES DEL LENGUAJE: TRADUCTORES, COMPILADORES E INTERPRETES-----	16
1.6.3 ENLAZADORES-----	17
1.6.4 DEPURADORES-----	17
CONCLUSION-----	18
BIBLIOGRAFIA-----	19

INTRODUCCION

La programación ha acompañado al ser humano desde la creación de la primera maquina funcional; a lo largo de la historia, esta se ha ido desarrollando cada vez mas en pro de los avances en materia tecnológica; desde su creación ideada especialmente para agilizar y reducir los costos operativos de las maquinas creadas en el “Siglo de las luces”, esta no se ha desviado del propósito principal, la eficacia.

En años posteriores a esta, la programación fue tras uno de los retos mas grandes para el ser humano, la comodidad. Esto se puede ver reflejado en nuestra era en múltiples objetos que nos rodean y usamos de manera cotidiana; Desde el despertador de nuestro smartphone, hasta la computadora que se usó para escribir esta investigación.

Esta investigación esta dedicada a todo aquel que desee informarse un poco más acerca de esta rama de la ciencia.

A lo largo de esta breve, pero completa investigación, me di a la tarea de recopilar y explicar, algunos de los eventos mas importantes que ha sufrido la programación como materia de estudio, así como sus bases y funcionalidades con las que cuenta en este momento.

Espero que lo disfrutes y puedas profundizar más en este ámbito de estudio y rama de las ciencias de la computación, llamado programación

1.1 EVOLUCION DE LA PROGRAMACION

1.1.1 ¿QUÉ ES LA PROGRAMACIÓN?

Según la Real Academia Española, Programar [1] se define como “preparar ciertas maquinas o aparatos para que empiecen a funcionar en el momento y en la forma deseados”, esto no se aleja de la realidad, puesto que la programación no comenzó tal y como nosotros interpretamos el concepto en la era moderna.

La programación ha acompañado al ser humano desde tiempos inmemoriales; para ser exactos, desde el siglo XVIII (mejor conocido como “El Siglo de las Luces”), surgieron grandes descubrimientos en favor de la humanidad, tales como el automóvil, el piano, el pararrayos [2], la invención de las máquinas de vapor y el milagro de la ciencia; la electricidad.

Con la llegada de inventos cada vez más complejos, los cuales suplían la gran cantidad de problemas que enfrentaba la humanidad en ese momento, surgía cada vez mas la necesidad de facilitar y agilizar los procesos analógicos de la recién llegada Revolución Industrial, época en la cual llegaron nuevos inventos como el telégrafo, los trenes de vapor, el reloj entre otros, cuyo funcionamiento solamente se acomodaba con el uso de palancas e interruptores conectados a mecanismos eléctricos.

Entre tantos grandes inventos, hubo uno el cual fue la base fundamental de lo que hoy conocemos como computadora. En [3] conocemos los múltiples conceptos de computadora; “Calculadora”, “Máquina que almacena y trata la información”, entre otros. Fue en 1837, cuando el matemático estadounidense Charles Babbage, decidió crear una maquina capaz de hacer cálculos de manera automática con tan solo accionar algunas palancas; La Máquina analítica [4].

La máquina analítica se describía como; “De naturaleza mecánica, incluía la mayoría de las partes lógicas de un ordenador actual. Era capaz de almacenar 1000 números de 50 dígitos cada uno” [5] todo esto con el accionar de palancas e introduciendo pequeños cilindros los cuales fungían de periféricos de entrada. Debido a la escasa comprensión de la tecnología en el momento, y a la falta de inversión para llevar a cabo su construcción, el desarrollo no se completó hasta 100 años después de su idealización.

Figura 1

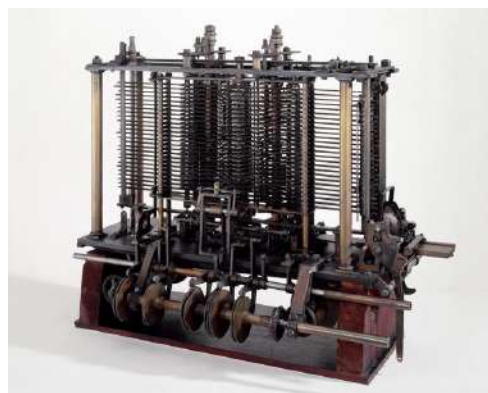
El automóvil



Fotografía del primer automóvil del mundo, adaptado de R.A(2021) *¿Sabias que el primer automóvil del mundo fue un Mercedes-Benz?* [Imagen] Rob Repport <https://robbreport.mx/motor/sabias-que-el-primer-automovil-del-mundo-fue-un-mercedes-benz-descubre-como-era-y-cuando-surgio/>

Figura 2

La máquina analítica



Fotografía de la maquina analítica, adaptado de C.T.L(2015) *De la maquina al PC en 10 pasos* [imagen] Cultura Científica <https://culturacientifica.com/2015/06/09/de-la-maquina-analitica-al-pc-en-10-pasos/>

1.1.2 EL PRIMER PROGRAMADOR DE LA HISTORIA

Augusta Ada King (1815-1852), mejor conocida como Ada Lovelace (*figura 3*), fue una matemática británica. Se considera fundadora de la programación como la conocemos hoy en día; escribió las normas para la maquina analítica de Babbage, que aún no había sido construida. Dedujo y previo la capacidad de los ordenadores para ir más allá de ser simples calculadoras numéricas, incluso mucho mas de lo que el propio Babbage pudo idealizar.

Ada fue la primera persona en describir un lenguaje de programación de carácter general al interpretar las ideas de Babbage. En 1843 publico algunas notas sobre su maquina analítica, las cuales firmó con sus iniciales, debido al miedo de ser censurada por ser mujer.

Ada escribió un completo plan donde se describe el algoritmo necesario para calcular los valores de los números de Bernoulli utilizando bucles; describió como realizar operaciones trigonométricas utilizando el termino de variable y definió el uso de tarjetas perforadas para programar las instrucciones de la máquina de Babbage [6].

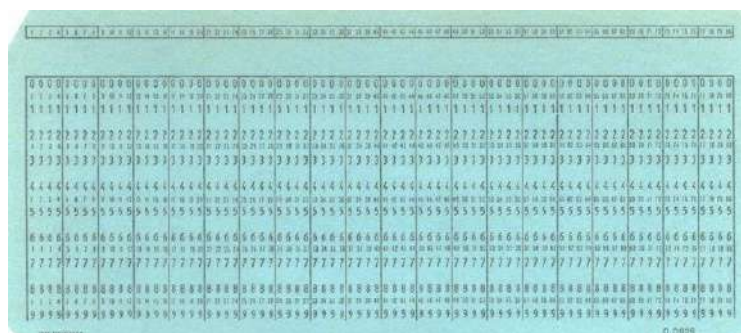
1.1.3 LA TARJETA PERFORADA

Como se describe en [7], el uso de tarjetas de lámina o cartón (*figura 4*), fue imprescindible para el funcionamiento de maquinas analógicas; Desde el telar automático de Joseph Marie Jacquard, inspirado en las cintas de las cajas musicales de Basile Bouchon, hasta la creación de las instrucciones y algoritmos de la maquina analítica, descritos por Luigi Menabrea y posteriormente actualizado y traducido al idioma inglés por Ada Lovelace.

La tarjeta perforada, es una lamina de cartulina la cual contiene información en forma de perforaciones según un código binario. Fueron los primeros medios utilizados para ingresar información e instrucciones a computadoras en los años 1960 y 1970.

Figura 4

Tarjeta perforada de cartón



Una tarjeta perforada típica para guardar datos en blanco, adaptado de C.C(2005) *Tarjeta perforada* [Imagen] Wikipedia https://es.wikipedia.org/wiki/Tarjeta_perforada

Figura 3

Ada Lovelace



Retrato de Ada Lovelace, adaptado de C.C(2020) *Ada Lovelace, la visionaria hija de Lord Byron* [Imagen] National Geographic

https://historia.nationalgeographic.com.es/a/ada-lovelace-visionaria-hija-lord-byron_15864

1.1.4 PROGRAMACION MAS CERCANA A LA ACTUALIDAD

Con la llegada de nuevas innovaciones en el ámbito tecnológico, la creación de los bulbos al vacío y las cintas magnéticas surgió la Primera Generación de Computadoras. No obstante, el tamaño de estas computadoras abarcaba el de una casa entera, su procesamiento era lento y su operación era muy ineficaz. No fue sino hasta 1941, cuando el ingeniero Konrad Zuse, diseñó lo que podríamos llamar “la primera computadora en el mundo que era programable de manera libre, basado en el sistema numerario binario y en tecnología de interruptores binarios” [8], la Z3 (figura 5).

En 1949, aparece el lenguaje Ensamblador (Assembly language), el consiste en un conjunto de mnemónicos que representan instrucciones básicas para los computadores, microprocesadores, microcontroladores y circuitos integrados programables. Constituye la representación más directa del código máquina [9].

En 1957 aparece FORTRAN, considerado el lenguaje de programación más antiguo que se utiliza en la actualidad. Creado para realizar cálculos científicos, matemáticos y estadísticos de alto nivel. Sigue utilizándose en los superordenadores más avanzados del mundo.

En 1964 BASIC (Beginners All-purpose Symbolic Instruction Code), fue desarrollado por estudiantes del Dartmouth College. Fue escrito para estudiantes que no tenían grandes conocimientos de matemáticas e informática. Bill Gates y Paul Allen, desarrollaron el lenguaje y lo convirtieron en el primer producto comercializable de su empresa, Microsoft.

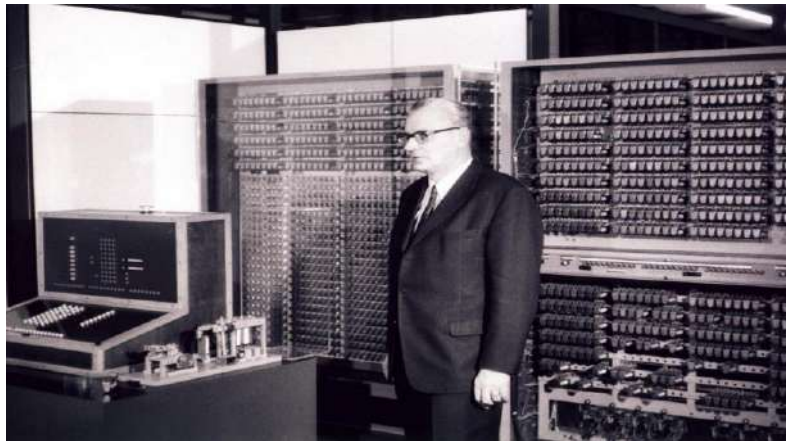
En 1972, nace C, desarrollado por Dennis Ritchie para su uso en el sistema operativo UNIX. Se llama C, porque se basaba en un lenguaje anterior llamado B. Muchos de los lenguajes actuales son derivados de C, como C#, Java, JavaScript, Perl, PHP y Python.

Con la llegada de 1983, C++ nace en los laboratorios Bell, siendo este una extensión de C, con mejoras en las clases, funciones virtuales y plantillas.

En 2014 Swift, desarrollado por Apple, llega como sustituto de C, C++ y ObjectiveC (lenguaje nativo de los dispositivos de esta empresa), para ser más sencillo y permitir menos margen de error en sus programas.

Figura 5

Konrad Zuse y la Z3



Konrad Zuse y la Z3 al fondo, adaptado de M.L.(2016) *La computadora Z3 de Konrad Zuse* [Imagen] <https://parceladigital.com/articulo/la-maquina-z3-de-zuse>

1.2 CONCEPTOS FUNDAMENTALES DE LA PROGRAMACION ORIENTADA A OBJETOS

¿QUE ES LA PROGRAMACION ORIENTADA A OBJETOS?

Según IBM, “La programación orientada a objetos se basa en el concepto de crear un modelo del problema de destino en sus programas. La programación orientada a objetos disminuye los errores y promueve la reutilización del código” [10]. Sin embargo, según Miriam M. Canelo, mánager en Profile [11] nos explica que la Programación Orientada a Objetos (POO por sus siglas en español) es un paradigma de programación, es decir, un modelo o estilo de programación que nos da unas guías para trabajar en él.

Siendo que lenguajes secuenciales como COBOL o procedimentales como Basic o C, se centran más en la lógica que en los datos. Otros mas modernos como Java, C# y Python, utilizan paradigmas para definir los programas, siendo la POO la más popular.

La forma de crear programas orientado a objetos consiste en hacer clases y crear objetos a partir de estas clases. Las clases forman el modelo a partir del cual se estructuran los datos y los comportamientos, lo cual veremos más adelante.

En la POO existen cuatro principios fundamentales a seguir

1.2.1 PRINCIPIO DE ENCAPSULACION

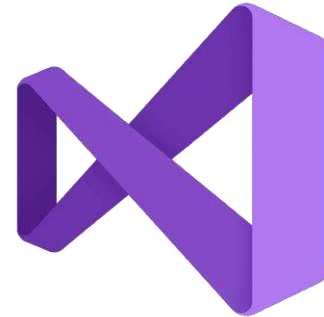
El principio de encapsulación contiene toda la información importante de un objeto dentro del mismo y solo expone la información seleccionada al mundo exterior (*figura 7*).

Esta propiedad permite asegurar que la información de un objeto este oculta para el mundo exterior, agrupando en una Clase las características o atributos que cuentan con un acceso privado y los comportamientos o métodos que presentan un acceso público. La encapsulación de cada objeto es responsable de su propia información y de su propio estado.

La única forma en la que este se puede modificar es mediante los propios métodos del objeto. Por lo tanto, los atributos internos de un objeto deberían ser inaccesibles desde fuera, pudiéndose modificar solo llamando a las funciones correspondientes. Con esto conseguimos mantener el estado a salvo de usos indebidos o que puedan resultar inesperados [11].

Figura 6

Logo de Visual Studio 2020



Logo de Visual Studio 2020, entorno de desarrollo orientado a objetos, adaptado de CC (2022), 1000 Marcas [Imagen] Visual Studio logo <https://1000marcas.net/visual-studio-logo/>

Figura 7

Cápsula



Imagen representativa del principio de encapsulación, adaptado de CC (2016), Icono de cápsula [Imagen] Vexel <https://es.vexels.com/png-svg/vista-previa/135594/icono-de-capsula>

1.2.2 PRINCIPIO DE ABSTRACCION

La abstracción es cuando el usuario interactúa solo con los atributos y métodos seleccionados de un objeto, utilizando herramientas simplificadas de alto nivel para acceder a un objeto complejo. En la programación orientada a objetos, los programas suelen ser muy grandes y los objetos se comunican mucho entre sí. El concepto de abstracción facilita el mantenimiento de un código de gran tamaño, donde a lo largo del tiempo pueden surgir diferentes cambios. Así, la abstracción se basa en usar cosas simples para representar la complejidad (*figura 8*).

Los objetos y las clases representan código subyacente, ocultando los detalles complejos al usuario. Por consiguiente, supone una extensión de la encapsulación [11].

Figura 8

Paisaje abstracto



Paisaje abstracto de un pequeño poblado, adaptado de CC (2022), *Paisaje abstracto* [Imagen] <https://www.royaltalens.com/es/inspiracion/planes-paso-a-paso/acrilico/paisaje-abstracto/>

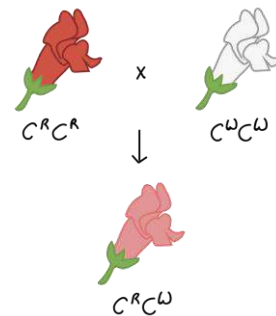
1.2.3 PRINCIPIO DE HERENCIA

La herencia define relaciones jerárquicas entre clases, de forma que atributos y métodos comunes puedan ser reutilizados. Las clases principales extienden atributos y comportamientos a las clases secundarias (*figura 9*).

A través de la definición en una clase de los atributos y comportamientos básicos, se pueden crear clases secundarias, ampliando así la funcionalidad de la clase principal y agregando atributos y comportamientos adicionales [11].

Figura 9

Herencia de características



Ejemplo de herencia de características en el ámbito genético, adaptado de CC (2015), *Variación que involucran genes individuales* [Imagen] <https://es.khanacademy.org/science/>

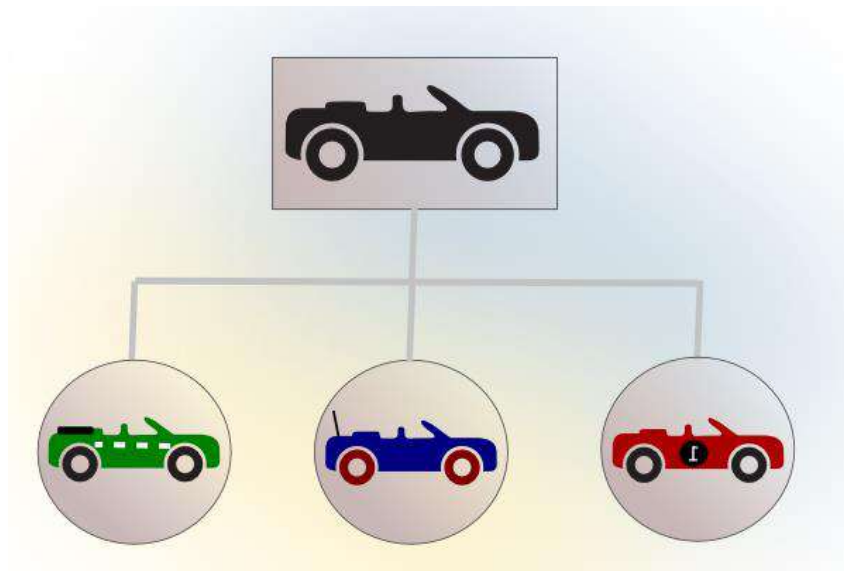
1.2.4 PRINCIPIO DE POLIMORFISMO

Consiste en diseñar objetos para compartir comportamientos, lo que nos permite procesar objetos de diferentes maneras. Es la capacidad de presentar la misma interfaz para diferentes formas subyacentes o tipos de datos (*figura 10*).

Al utilizar la herencia, los objetos pueden anular los comportamientos principales compartidos, con comportamientos secundarios específicos. El polimorfismo permite que el mismo método ejecute diferentes comportamientos de dos formas: anulación de método y sobrecarga de método [11].

Figura 10

Polimorfismo



Ejemplo de polimorfismo, adaptado de CC (2021), *Polimorfismo en Java* [Imagen]
<http://www.buscaminegocio.com/cursos-de-java/polimorfismo-java.html>

1.3 LENGUAJE ORIENTADO A OBJETOS

Se le llama así a cualquier lenguaje de programación que implemente los conceptos definidos por la programación orientada a objetos. Cabe notar que los conceptos definidos en la programación orientada a objetos no son una condición sino que son para definir que un lenguaje es orientado a objetos. Existen conceptos que pueden estar ausentes en un lenguaje dado y, sin embargo, no invalidar su definición como lenguaje orientado a objetos. Quizás las condiciones mínimas necesarias las provee el formalismo que modeliza mejor las propiedades de un sistema orientado a objetos: los tipos de datos abstractos. Siguiendo esa idea, cualquier lenguaje que permita la definición de tipos de datos, de operaciones nuevas sobre esos tipos de datos, y de instanciar el tipo de datos podría ser considerado orientado a objetos [12].

Los lenguajes de programación orientados a objetos son lenguajes dinámicos en los que estos objetos se pueden crear y modificar sobre la marcha. Esta programación orientada a objetos (POO) tuvo auge a mediados de los años ochenta debido a la propagación de las interfaces gráficas de usuarios, para lo que los lenguajes de programación orientados a objetos están especialmente dotados. Los principales lenguajes de programación orientados a objetos son: C++, C#, VB.NET, Java, ObjectiveC, y Python (figura 11) [13].

Figura 11

Lenguajes de programación orientado a objetos



C++



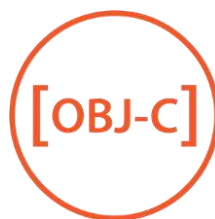
C#



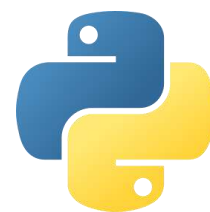
Visual Studio .NET



Java



ObjectiveC



Python

Ejemplo de lenguajes de programación orientados a objetos, J.D (2022) *Algunos lenguajes de programación orientado a objetos* [Clip Art] Microsoft Office Word 2021

1.4 RELACION ENTRE CLASES Y OBJETOS

¿Cómo se crean los programas orientados a objetos? Resumiendo, consistiría en hacer clases y crear objetos a partir de estas clases. Las clases forman el modelo a partir del que se estructuran los datos y los comportamientos. El primer y más importante concepto de la POO es la distinción entre clase y objeto. Una clase es una plantilla. Define de manera genérica cómo van a ser los objetos de un determinado tipo. Por ejemplo, una clase para representar a animales puede llamarse 'animal' y tener una serie de atributos, como 'nombre' o 'edad' (que normalmente son propiedades), y una serie con los comportamientos que estos pueden tener, como caminar o comer, y que a su vez se implementan como métodos de la clase (funciones).

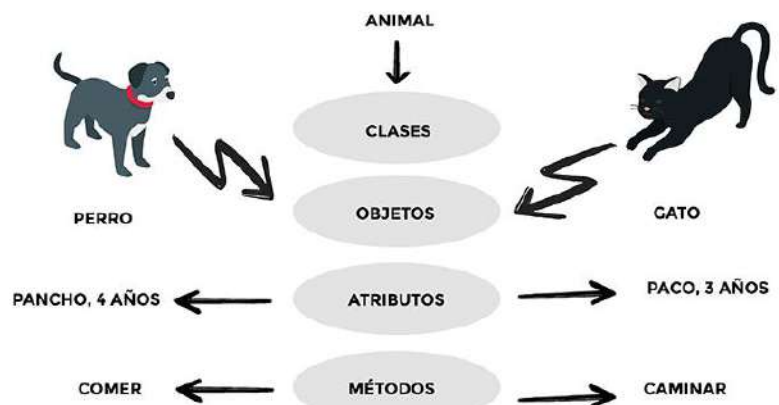
Un ejemplo sencillo de un objeto, como decíamos antes, podría ser un animal. Un animal tiene una edad, por lo que creamos un nuevo atributo de 'edad' y, además, puede envejecer, por lo que definimos un nuevo método. Datos y lógica. Esto es lo que se define en muchos programas como la definición de una clase, que es la definición global y genérica de muchos objetos [11].

Figura 12

Con la clase se pueden crear instancias de un objeto, cada uno de ellos con sus atributos definidos de forma independiente. Con esto podríamos crear un gato llamado Paco, con 3 años, y otro animal, este tipo perro y llamado Pancho, con una edad de 4 años (figura 12).

Los dos están definidos por la clase animal, pero son dos instancias distintas. Por lo tanto, llamar a sus métodos puede tener resultados diferentes. Los dos comparten la lógica, pero cada uno tiene su estado de forma independiente [11].

Ejemplo de clases, objetos, atributos y métodos



 profile

Ejemplo de objeto (gato paco). M.C (2020) ¿Qué es la programación orientada a objetos [Imagen] Profile <https://profile.es/blog/que-es-la-programacion-orientada-a-objetos/>

1.5 PAPEL DE CLASE Y OBJETO EN EL ANÁLISIS Y DISEÑO

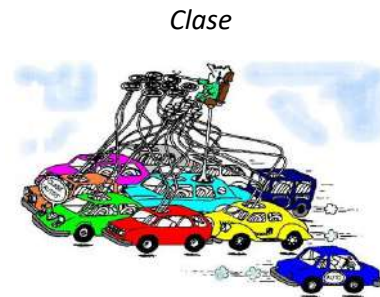
Durante el análisis y las primeras etapas del diseño, el desarrollador tiene dos tareas principales: Identificar las clases y objetos que forman el vocabulario del dominio del problema.

Idear las estructuras por las que conjuntos de objetos trabajan juntos para lograr los comportamientos que satisfacen los requerimientos del problema.

El análisis orientado a objetos es un método de análisis que examina los requisitos desde la perspectiva de las clases y objetos que se encuentran en el vocabulario del dominio del problema [14].

Un objeto tiene estado, exhibe algún comportamiento bien definido, tiene una identidad única. Una clase representa un conjunto de objetos que comparten una estructura y un comportamiento comunes [15] (figura 13).

Figura 13



1.5.1 ELEMENTOS DEL MODELO DE OBJETOS

1.5.1.1 MODULARIDAD

Es la propiedad que tiene un sistema que ha sido descompuesto en un conjunto de módulos cohesivos y débilmente acoplados (figura 14) [15].

1.5.1.2 JERARQUÍA

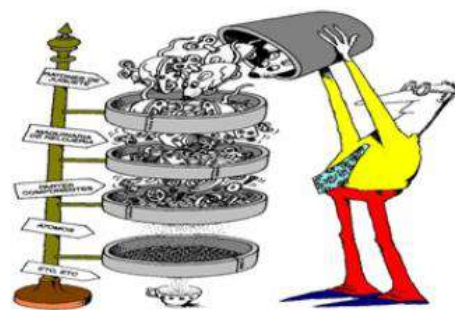
Es una clasificación u ordenación de abstracciones (figura 14) [15].

Figura 14

Elementos fundamentales



MODULARIDAD



JERARQUÍA

Ejemplo de modularidad y jerarquía M.H (2014), *Análisis y Diseño orientado a objetos* [Clip art] SlidePlayer <https://slideplayer.es/slide/94222/>

1.5.1.3 TIPIFICACION

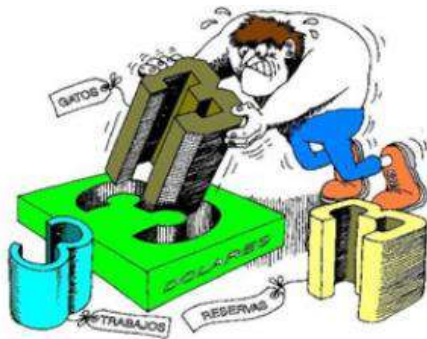
Son la puesta en vigor de la clase de los objetos, de forma que los objetos de tipos diferentes no pueden intercambiarse, o pueden hacerlo de forma restringida (*figura 15*) [15].

1.5.1.4 CONCURRENCIA

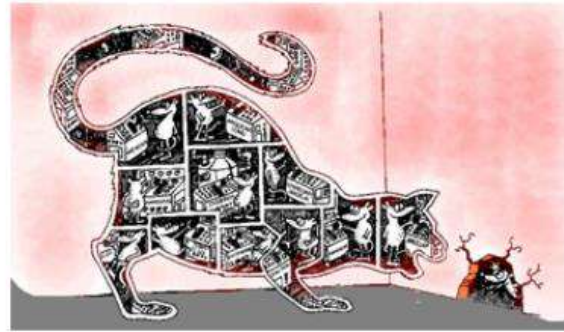
Es la propiedad que distingue un objeto activo, de uno que no esta activo. Permite manejar muchos eventos diferentes a la vez (*figura 15*) [15].

Figura 15

Elementos fundamentales



TIPIFICACIÓN



CONCURRENCIA

Ejemplo de tipificación y concurrencia M.H (2014), *Análisis y Diseño orientado a objetos* [Clip art] SlidePlayer <https://slideplayer.es/slide/94222/>

1.6 ENTORNOS DE PROGRAMACION

Un entorno de programación es un programa o conjunto de programas que engloban todas las tareas necesarias para el desarrollo de un programa o aplicación. Estas tareas son básicamente las siguientes:

- Edición del programa.
- Compilación y enlazado.
- Ejecución.
- Depuración.

Hay quien además incluye la creación de documentación complementaria que facilita el mantenimiento del programa dentro de estas funciones.

Este tipo de entornos incorporan numerosas herramientas, utilidades, aplicaciones ya desarrolladas, ejemplos, tutoriales, etc. Todas ellas encaminadas a facilitar y mejorar el desarrollo (*figura 16*) [16].

1.6.1 EDITORES DE TEXTO

El primer elemento necesario para el desarrollo de un programa es un editor de texto. Un editor es un programa que nos permite escribir (editar) las instrucciones del programa y posteriormente guardar el programa en un fichero en un soporte de almacenamiento. Cualquier editor de texto se puede utilizar para editar programas con la única precaución de que, a la hora de guardar, salvar o almacenar el programa sólo se almacene el texto sin opciones de formato: negrita, estilos, itálica, etc. (*figura 17*).

Lo normal es utilizar un editor especialmente preparado para la programación. Estos tienen facilidades para la corrección de errores, destacan las palabras del lenguaje en colores, y en general facilitan la labor del programador [16].

Figura 16



Logo de Eclipse 2020, entorno de desarrollo, adaptado de CC (2022), 1000 Marcas [Imagen] Eclipse logo <https://1000marcas.net/eclipse-logo/>

Figura 17



Logo de Notepad++ 2020, entorno de desarrollo, adaptado de CC (2022), 1000 Marcas [Imagen] Notepad++e logo <https://1000marcas.net/notepad-logo/>

1.6.2 PROCESADORES DEL LENGUAJE: TRADUCTORES, COMPILADORES E INTERPRETES.

Una vez editado nuestro programa es necesario que este sea procesado y transformado en órdenes que puedan ser ejecutadas por el ordenador. Estas órdenes por tanto deben estar en el único lenguaje que la máquina entiende: el código máquina. Para ello son necesarios los procesadores de lenguaje cuyo concepto es muy amplio. Dentro de los procesadores de lenguaje destacan los traductores, los compiladores y los intérpretes.

Un compilador es un programa cuyo cometido es realizar la conversión de un programa escrito en un lenguaje de programación a su correspondiente equivalente en lenguaje máquina. El resultado que devuelve un compilador es un programa que ya puede ser ejecutado por el ordenador destino sin la necesidad de que el compilador esté presente. Por ejemplo, el lenguaje Pascal o el lenguaje C son lenguajes de programación que necesitan ser compilados. Cuando la conversión se realiza entre el lenguaje ensamblador (*Assembly*) y el código máquina, el compilador recibe el nombre específico de *Ensamblador (Assembler)* (figura 18).

Figura 18

GNU, compilador libre



Logo de GNU 2020, compilador de LINUX, adaptado de CC (2022),
1000 Marcas [Imagen] GNU logo <https://1000marcas.net/gnu-logo/>

Un intérprete es un programa que convierte línea por línea el programa escrito en un lenguaje de programación y que a medida que realiza la conversión ejecuta las instrucciones. Evidentemente el intérprete no devuelve nada ya que la ejecución se realiza de forma simultánea. Por este motivo, el intérprete debe estar presente durante la ejecución. Lenguajes de programación que tradicionalmente son interpretados son el LISP y el BASIC.

Un traductor es el nombre que reciben aquellos procesadores de lenguaje que convierten programas de unos lenguajes a otros, pero no generan código máquina. Por ejemplo, hay traductores de Pascal a C y viceversa.

Hay otros lenguajes de programación que combinan ambas estrategias como por ejemplo sucede con el lenguaje de programación Java. Para este lenguaje existen traductores que generan un programa en un código denominado intermedio que luego será ejecutado a través de un intérprete que recibe en este caso el nombre de máquina virtual Java.

1.6.3 ENLAZADORES

Por simplificación y para facilitar la comprensión de los conceptos anteriores se ha señalado que los compiladores y los ensambladores (caso particular de compilador) generan código máquina que puede ser ejecutado por el ordenador. Sin embargo, esto no es totalmente cierto ya que hay una etapa de enlazado que debe ser realizada por otro programa denominado enlazador (*linker*). Lo habitual es que durante la escritura de un programa sea necesario utilizar otros subprogramas en forma de bibliotecas de funciones o bien que el propio programa esté formado realmente por varios programas almacenados en diferentes ficheros. Esta situación hace que durante la compilación de cada módulo no se conozca con exactitud la ubicación de las instrucciones del resto de programas o bibliotecas de funciones. El papel del enlazador es unir en un único fichero ejecutable el resultado de todas las compilaciones, así como las bibliotecas estáticas de funciones. Es frecuente que el enlazado sea un paso más de la compilación y que se ejecute inmediatamente tras la compilación de todos los ficheros [16].

Es habitual denominar a cada uno de los ficheros que participan en el desarrollo de un programa con nombres genéricos que identifican en qué fase se encuentran. Por ejemplo, las instrucciones que escribe directamente el programador y que forman el programa en el lenguaje de programación escogido como ficheros fuente (*source file*). El resultado de la compilación de estos programas se denomina fichero objeto (*object file*) y por el último el resultado del enlazado fichero ejecutable (*executable file*). Es este último el único que puede entender un ordenador sin la presencia del compilador. En el caso de los lenguajes interpretados el fichero fuente es directamente interpretado y ejecutado por el intérprete.

1.6.4 DEPURADORES

Una vez editado y compilado el programa es necesario ejecutarlo (*run* en inglés), pero es habitual que durante el desarrollo de una aplicación que generen ficheros ejecutables que, aunque sean correctos desde un punto de vista sintáctico no realicen lo que realmente se espera de ellos por lo que se consideran que no funcionan correctamente.

Los depuradores (*debuggers*) son capaces de ejecutar el programa paso a paso incluyendo además un conjunto de facilidades que permiten observar el valor de las variables y estructuras de datos permitiendo así una mejor localización de errores no evidentes [16].

CONCLUSION

Como hemos observado a través de este breve viaje en el tiempo, la programación ha sufrido cambios muy significativos como dejar de utilizar tarjetas de cartón, a usar un método de entrada como lo es el teclado. Y otros muy curiosos como que la programación se planeó para crear tejidos de manera automática con poca o nula interacción humana.

El propósito de esta investigación fue introducirte principalmente en la programación orientada a objetos, la cual además de ser muy fácil de utilizar, podría decirse que nos rodea en todos lados; por ejemplo, en los navegadores, en los programas que usamos para editar texto (procesadores de texto) e incluso en aquellos programas que usamos para divertirnos después de un largo día de trabajo como lo son los videojuegos.

Espero que mi obra te haya servido para convencerte a entrar a este maravilloso mundo, donde;

“Si puedes imaginarlo, puedes programarlo”

Alejandro Taboada Sánchez

(1997-2019).

BIBLIOGRAFIA

- [1] "Programar | Diccionario de la lengua española". «Diccionario de la lengua española» - Edición del Tricentenario. <https://dle.rae.es/programar> (accedido el 18 de septiembre de 2022).
- [2] G. Lp y P. Gs. "Invenciones del Siglo XVIII". Sutori. <https://www.sutori.com/es/historia/inventos-de-los-siglos-xviii-xix--6LTcVcERs497w4BsgAkjJ3dy> (accedido el 18 de septiembre de 2022).
- [3] "Computador, computadora | Diccionario de la lengua española". «Diccionario de la lengua española» - Edición del Tricentenario. <https://dle.rae.es/computador> (accedido el 18 de septiembre de 2022).
- [4] Colaboradores de los proyectos Wikimedia. "Máquina analítica - Wikipedia, la enciclopedia libre". Wikipedia, la enciclopedia libre. https://es.wikipedia.org/wiki/Máquina_analítica (accedido el 18 de septiembre de 2022).
- [5] "La máquina analítica de babbage | GTD blog". Ingeniería de sistemas y software: Soluciones TIC a la medida. <https://www.gtd.es/es/blog/la-maquina-analitica-de-babbage> (accedido el 18 de septiembre de 2022).
- [6] "Ada Lovelace, primer programador en la historia | News+Media". Dalia Empower. <https://daliaempower.com/blog/ada-lovelace-primer-programador-en-la-historia> (accedido el 18 de septiembre de 2022).
- [7] Colaboradores de los proyectos Wikimedia. "Tarjeta perforada - Wikipedia, la enciclopedia libre". Wikipedia, la enciclopedia libre. https://es.wikipedia.org/wiki/Tarjeta_perforada (accedido el 18 de septiembre de 2022).
- [8] T. Lp. "La generación Z: Konrad Zuse, pionero alemán de la computación". Centro Alemán de información para Latinoamérica - Ministerio Federal de Relaciones Exteriores. <https://alemaniaparati.diplo.de/mxdz-es/aktuelles/konradzuse/1087764#:~:text=Zuse%20la%20define%20como%20%20la,realizar%20las%20operaciones%20matemáticas%20básicas.> (accedido el 18 de septiembre de 2022).
- [9] Colaboradores de los proyectos Wikimedia. "Lenguaje ensamblador - Wikipedia, la enciclopedia libre". Wikipedia, la enciclopedia libre. https://es.wikipedia.org/wiki/Lenguaje_ensamblador (accedido el 18 de septiembre de 2022).
- [10] "IBM documentation". IBM - Deutschland | IBM. <https://www.ibm.com/docs/es/spss-modeler/saas?topic=language-object-oriented-programming> (accedido el 18 de septiembre de 2022).
- [11] M. M Canelo. "¿Qué es la programación orientada a objetos?" Profile Software Services. <https://profile.es/blog/que-es-la-programacion-orientada-a-objetos/> (accedido el 18 de septiembre de 2022).
- [12] Colaboradores de los proyectos Wikimedia. "Lenguaje orientado a objetos - Wikipedia, la enciclopedia libre". Wikipedia, la enciclopedia libre. https://es.wikipedia.org/wiki/Lenguaje_orientado_a_objetos (accedido el 18 de septiembre de 2022).
- [13] "Lenguajes de programación orientada a objetos". La revista informática. <http://www.larevistainformatica.com/lenguajes-programacion-orientada-objetos.htm> (accedido el 18 de septiembre de 2022).
- [14] J. D. Muñoz Martínez, *Papel de las clases y objetos*. 2013. Accedido el 18 de septiembre de 2022. [En línea]. Disponible: https://nanopdf.com/download/clases-objetos_pdf
- [15] M. Hidalgo. "METODOLOGÍA ORIENTADA A OBJETOS CARACTERÍSTICAS DEL PROCESO - ppt video online descargar". SlidePlayer - descarguen y compartan sus presentaciones PowerPoint. <https://slideplayer.es/slide/94222/> (accedido el 18 de septiembre de 2022).
- [16] "Lenguajes de programación: Entornos de programación". http://descargas.pntic.mec.es/mentor/visitas/nav_Inici_Progr/lenguajes/len20.html#:~:text=Un%20entorno%20de%20programación%20es,Compilación%20y%20enlazado. (accedido el 18 de septiembre de 2022).



INSTITUTO TECNOLÓGICO SUPERIOR DE SAN ANDRÉS TUXTLA



ALGORITMOS

PLAN DE ESTUDIO: INGENIERIA INFORMATICA

ALUMNO: EMMANUEL DE JESUS TEOBAL DIAZ

DOCENTE: ENRIQUE TELONA TORRES

18/09/2022

INTRODUCCION

Mediante la presente investigación, se pretende dar conocer, entender y reproducir el termino algoritmo. Este proyecto esta dirigido para aquel publico cuyo conocimiento sobre la programación es demasiado superficial y desee profundizar en uno de los temas mas fundamentales sobre la programación.

Así mismo, se explica las características del entorno de desarrollo PSEINT, el cual, durante el transcurso de la investigación, tendrá cada vez más desarrollo. En este proyecto analizaremos las características de un algoritmo, la definición de Constantes, Variables, identificadores y Operadores en el entorno PSEINT.

Mas adelante se presentará las seis fases de resolución de problemas, para posteriormente aplicarlo en nuestros proyectos y adaptarlo para cualquier problema en el ámbito de la programación en general.

DEFINICION DE ALGORITMO

En matemáticas, lógica, ciencias de la computación y disciplinas relacionadas, un algoritmo es un conjunto de instrucciones o reglas definidas y no-ambiguas, ordenadas y finitas que permite, típicamente, solucionar un problema, realizar un cómputo, procesar datos y llevar a cabo otras tareas o actividades [1].

Según los expertos en matemática, los algoritmos permiten trabajar a partir de un estado básico o inicial y, tras seguir los pasos propuestos, llegar a una solución. Cabe resaltar que, si bien los algoritmos suelen estar asociados al ámbito matemático (ya que permiten, por citar casos concretos, averiguar el cociente entre un par de dígitos o determinar cuál es el máximo común divisor entre dos cifras pertenecientes al grupo de los enteros), aunque no siempre implican la presencia de números [2].

CARACTERISTICAS DE UN ALGORITMO

Un algoritmo debe de ser:

FINITO.

Es decir, debe finalizar tras un número determinado de pasos.

DEFINIDO.

El algoritmo debe de definirse de forma precisa para cada paso, es decir, debemos evitar cualquier ambigüedad, de modo que, si se sigue n veces, debemos obtener el mismo resultado cada vez.

PRECISO.

Todas las operaciones que el algoritmo realice deberán ser lo suficientemente básicas, de modo que puedan desde un principio, ser llevadas a cabo de manera exacta y en un tiempo finito por una persona usando lápiz y papel.

IDENTIFICADORES EN PSEINT

Los identificadores comúnmente llamados variables en programación, son aquellas palabras que elegimos para asignarles un valor. En PSEINT debemos seguir ciertas normas:

Los identificadores, o nombres de variables, deben constar sólo de letras, números y/o guion bajo (`_`), comenzando siempre con una letra. No puede contener símbolos (`@`, `#`, `▼`, `[,]`, `=`, etc.), ni eñes (`ñ` o `Ñ`) y no puede tener espacios en blancos [3].

Los identificadores, o nombres de variables, no pueden coincidir con las palabras reservadas del lenguaje. PSEINT colorea de azul Las palabras reservadas. Se pueden crear variables del mismo tipo en una solo líneas separadas por comas Definir `n1, n2, suma` Como Entero [3]

DIFERENCIA ENTRE VARIABLE Y CONSTANTE

La diferencia principal entre una constante y una variable es que las variables pueden cambiar de valor durante la ejecución del programa. Caso contrario a las constantes, que como su nombre lo indica, siguen siendo constante en su valor a lo largo de la ejecución.

¿QUE SON LOS OPERADORES?

En Matemáticas los operadores nos sirven para realizar la representación de fórmulas. En la Programación, los operadores nos sirven como para utilizarse en fórmulas dentro de un algoritmo además de realizar comparaciones y también para agrupar elementos. En el esquema del presente trabajo, veremos los tipos de operadores que se utilizan en la programación de forma general cuando se elabora un algoritmo. Ya que dependiendo del Lenguaje de Programación el símbolo varía [4].

OPERADORES EN PSEINT

A continuación, presento una serie de operadores de PSEINT (figura 1)

Figura 1

Tabla de operadores de PSEINT

Operador	Significado	Ejemplo
Relacionales		
>	Mayor que	3>2
<	Menor que	'ABC'<'abc'
=	Igual que	4=3
<=	Menor o igual que	'a'<='b'
>=	Mayor o igual que	4>=5
<>	Distinto que	Var1<>var2
Lógicos		
& ó Y	Conjunción (y).	(7>4) & (2=1) //falso
ó O	Disyunción (o).	(1=1 2=1) //verdadero
~ ó NO	Negación (no).	~(2<5) //falso
Algebraicos		
+	Suma	total <- cant1 + cant2
-	Resta	stock <- disp - venta
*	Multiplicación	area <- base * altura
/	División	porc <- 100 * parte / total
^	Potenciación	sup <- 3.41 * radio ^ 2
% ó MOD	Módulo (resto de la división entera)	resto <- num MOD div www.informeglobal.com

Tabla de operadores en PSEINT, CC (2017), *Expresiones en PSEINT: Operadores y Funciones Matemáticas*, adaptado de Informe Global [Imagen] <https://informeglobal.com/pseint-operadores-funciones-resumen/>

ETAPAS PARA LA RESOLUCION DE PROBLEMAS

ANALISIS

El primer paso para encontrar la solución a un problema es el análisis del mismo. Se debe examinar cuidadosamente el problema a fin de obtener una idea clara sobre lo que solicita y determinar lo que se necesita para conseguirlo. El problema se analiza teniendo en presente la especificación de los requisitos dados por la persona que requiere el programa (el usuario) [5].

DISEÑO

Una vez analizado el problema, se diseña una solución que conducirá a un *algoritmo* que resuelva el problema. Se plantean, la forma en que se reciben los datos de entrada, el proceso por el cual se produce el resultado y la forma en que se muestra la salida. Una vez que se ha terminado de escribir un algoritmo es necesario comprobar que realiza la tarea para la cual fue diseñado y produce el resultado correcto y esperado [5].

CODIFICACIÓN (IMPLEMENTACIÓN)

El algoritmo diseñado se escribe en la sintaxis del lenguaje de programación seleccionado, obteniendo así el programa fuente [5].

EJECUCIÓN, VERIFICACIÓN Y DEPURACIÓN

El programa se ejecuta, se comprueba rigurosamente y se eliminan todos los errores que puedan aparecer (eliminar *bugs*). Asumiendo una competente codificación, entre mejor se haga todo en las fases de análisis y diseño menos tiempo se gastará buscando, identificando y solucionando errores. Cuando se ejecuta un programa pueden producirse tres tipos de errores: *de traducción, de ejecución y lógicos* [5].

DOCUMENTACIÓN.

Documentos que soportan todo lo realizado con respecto al programa. Incluye diseño, codificación, manuales, normas, etc. Se recomienda ir documentando cada cosa que se hace, en vez de esperar hasta el final y tener que documentar todo de principio a fin. Existe documentación interna, incluida dentro del código fuente mediante *comentarios*, y documentación externa. La documentación es vital para que los usuarios puedan usar el programa y para que los programadores entiendan y modifiquen el código fuente [5].

MANTENIMIENTO.

El programa se actualiza y modifica cada vez que sea necesario, de modo que se cumplan todas las necesidades adicionales de los usuarios [5].

CONCLUSION

A lo largo de esta breve investigación, hemos podido informarnos acerca de la definición de un algoritmo, las características que debe poseer un algoritmo. También analizamos Las distintas características del entorno de desarrollo PSEINT, tales como operadores, identificadores, variables y constantes. Aprendimos cada una de las seis fases de la metodología para resolver problemas y como aplicarlo en nuestros trabajos como programadores. Espero que haya quedado claro la importancia de los algoritmos, ya que, sin irnos al medio tecnológico, día a día seguimos y desarrollamos algoritmos, algunos que van desde seguir los pasos de una receta, hasta seguir indicaciones para poder llegar a un lugar en específico.

BIBLIOGRAFIA

[1] Colaboradores de los proyectos Wikimedia. "Algoritmo – Wikipedia, la enciclopedia libre". Wikipedia, la enciclopedia libre. <https://es.wikipedia.org/wiki/Algoritmo> (accedido el 19 de septiembre de 2022).

[2] "Definición de algoritmo – 7definición.de". Definición.de. <https://definicion.de/algoritmo/> (accedido el 19 de septiembre de 2022).

[3] Consellería de Cultura, Educación e Universidade |. [https://www.edu.xunta.gal/centros/iesblancoamorculleredo/aulavirtual/pluginfile.php/37189/mod_assign/intro/ApuntesPSEINT201617%20\(1\).pdf](https://www.edu.xunta.gal/centros/iesblancoamorculleredo/aulavirtual/pluginfile.php/37189/mod_assign/intro/ApuntesPSEINT201617%20(1).pdf) (accedido el 19 de septiembre de 2022).

[4] "Operadores para algoritmos". Universidad Autónoma del Estado de Hidalgo: UAEH. <https://www.uaeh.edu.mx/scige/boletin/prepa2/n7/m2.html#:~:text=En%20la%20Programación,%20los%20operadores,cuando%20se%20elabora%20un%20algoritmo.> (accedido el 19 de septiembre de 2022).

[5] "3.1 resolución de problemas | 2016374 programación en lenguajes estadísticos". Home | Bookdown. <https://bookdown.org/cjtorresj/ple/resolución-de-problemas.html> (accedido el 19 de septiembre de 2022).



Emmanuel de Jesus Teobal Diaz

jesus.laboet.official@gmail.com

◀ Cambiar usuario ▼ ▶

30 de 34 ▼



◀ Página 1 de 1 ▶



Entrega

Enviado para calificar

Calificado

La tarea fue enviada 1 hora 35 minutos antes



El estudiante puede editar esta entrega

 [Unidad1_TeobalDiaz.zip](#)

► Comentarios (0)

Calificación

Calificación:

Hoja de presentación	No contien todos los datos 0 puntos	Datos incompletos 1 puntos	Completo 2 puntos	
Introducción	No contiene 0 puntos	Muy pequeña 2.5 puntos	Completa 5 puntos	

Contenido	No cubre los temas 0 puntos	La mitad de los temas 6 puntos	Completo 13 puntos	
Referencias IEEE	No contiene 0 puntos	Una o no tiene el formato 2 puntos	Más de una y formato correcto 4 puntos	
Conclusión	No contiene 0 puntos	Muy pequeña 2.5 puntos	Completa 5 puntos	
Archivo PDF	Sin formato 0 puntos	Correcto 1 puntos		

Calificación actual en el libro de calificaciones

30,00

Comentarios de retroalimentación



Notificar a los estudiantes



Guardar cambios

Reiniciar



INSTITUTO TECNOLÓGICO SUPERIOR DE SAN ANDRÉS TUXTLA



PRACTICAS DE PROGRAMACION

PLAN DE ESTUDIO: INGENIERIA INFORMATICA

ALUMNO: EMMANUEL DE JESUS TEOBAL DIAZ

DOCENTE: ENRIQUE TELONA TORRES

20/09/2022

INDICE

Practica 1	3
Practica 2	4
Practica 3	5
Practica 4	6
Practica 5	7
Practica 6	8
Practica 7	9
Practica 8	10
Practica 9	11
Practica 10	12
Practica 10.1	13
Practica 11	14
Conclusión	15

PRACTICA 1

The image shows a screenshot of a Pascal IDE. The main window displays a Pascal program named 'Ejercicio_1.psc'. The code is as follows:

```
1 Proceso sin_titulo
2   //Algoritmo que imprime texto
3   //Emmanuel de jesus Teobal Diaz
4
5   Escribir 'Fundamentos de Programacion';
6   Escribir 'Autor(es): Emmanuel de jesus & Santos de jesus ;)';
7   Escribir 'A 6 de Septiembre de 2022';
8 FinProceso
9
```

An execution window titled 'PSeInt - Ejecutando proceso SIN_TITULO' is overlaid on the code. It shows the output of the program:

```
*** Ejecución Iniciada. ***
Fundamentos de Programacion
Autor(es): Emmanuel de jesus & Santos de jesus ;)
A 6 de Septiembre de 2022
*** Ejecución Finalizada. ***
```

The execution window also has a 'Reiniciar' button and checkboxes for 'No cerrar esta ventana' (checked) and 'Siempre visible' (unchecked).

PRACTICA 2

The image shows the PSeInt software interface. The main window displays a Pascal program titled "Ejercicio_2.psc" with the following code:

```
1 Proceso sin_titulo
2
3 //Programa estatico suma 3+2
4 //Autor: Emmanuel de Jesus Teobal Diaz
5 //06 de septiembre del 2022
6
7
8 escribir 3+2;
9
10 FinProceso
11
```

An execution window titled "PSeInt - Ejecutando proceso SIN_TITULO" is open, showing the output of the program:

```
*** Ejecución Iniciada. ***
5
*** Ejecución Finalizada. ***
```

The execution window also includes checkboxes for "No cerrar esta ventana" and "Siempre visible", and a "Reiniciar" button.

The right side of the interface features a "Comandos" panel with various control flow symbols and their corresponding labels: "Escribir", "Leer", "Asignar", "Si-Entonces", "Según", "Mientras", "Repetir", "Para", and "SubProceso".

PRACTICA 3

The image shows the PSeInt IDE interface. The main window displays a Pascal program with the following code:

```
1 Proceso sin_titulo
2 //Este Algoritmo muestra en pantalla una suma de dos terminos N usando la concatenacion
3 //Autor: Emmanuel de Jesus Teobal Diaz
4 Definir num1,num2 como entero;
5
6 num1 ← 30;
7 num2 ← 5;
8
9 escribir num1, '+', num2, '=', num1+num2;
10 .....
11 FinProceso
12
```

An execution window titled "PSeInt - Ejecutando proceso SIN_TITULO" is open, showing the output of the program:

```
*** Ejecución Iniciada. ***
30+5=35
*** Ejecución Finalizada. ***
```

The execution window also includes checkboxes for "No cerrar esta ventana" (checked) and "Siempre visible" (unchecked), and a "Reiniciar" button.

The IDE interface includes a menu bar (Archivo, Editar, Configurar, Ejecutar, Ayuda), a toolbar with various icons, and a right-hand sidebar with a "Comandos" panel containing various programming constructs like "Hola", "Datos", "A ← B +", and "Si...Entonces...".

PRACTICA 4

The image shows the PSeInt IDE interface. The main window displays a Pascal program titled "Proceso sin_titulo" with the following code:

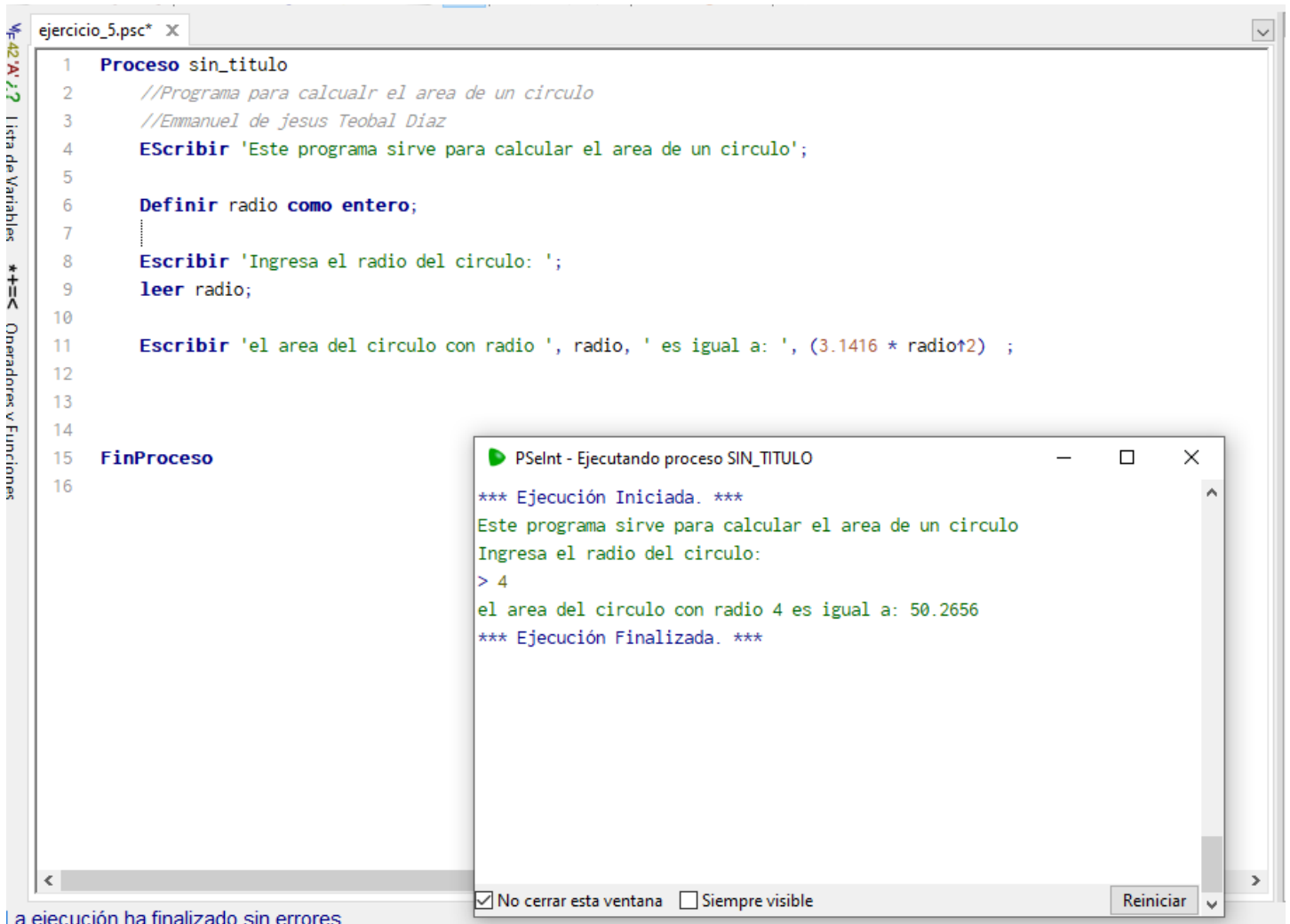
```
1 Proceso sin_titulo
2 //Este algoritmo suma dos numeros ingresados por el usuario, al final lo muestra en pantalla con concatenacion
3 //Emmanuel de Jesus Teobal Diaz
4 definir num1, num2 como entero;
5
6 Escribir 'Programa para sumar 2 numeros';
7 Escribir 'Ingresa el primer Numero: ';
8 leer num1;
9 escribir 'Ingresa el segundo Numero: ';
10 leer num2;
11
12 definir suma como entero;
13 suma ← num1 + num2;
14
15 escribir 'La suma de ', num1, ' + ', num2, ' es igual a: ', suma ;
16
17
18 FinProceso
19
20
```

An execution window titled "PSeInt - Ejecutando proceso SIN_TITULO" is overlaid on the code, showing the following output:

```
*** Ejecución Iniciada. ***
Programa para sumar 2 numeros
Ingresa el primer Numero:
> 34
Ingresa el segundo Numero:
> 65
La suma de 34 + 65 es igual a: 99
*** Ejecución Finalizada. ***
```

At the bottom of the IDE, a status bar indicates "La ejecución ha finalizado sin errores." and a control bar contains the options "No cerrar esta ventana" (checked), "Siempre visible" (unchecked), and a "Reiniciar" button.

PRACTICA 5



The image shows a Pascal IDE window titled 'ejercicio_5.psc'. The code is as follows:

```
1 Proceso sin_titulo
2 //Programa para calculr el area de un circulo
3 //Emmanuel de Jesus Teobal Diaz
4 Escribir 'Este programa sirve para calcular el area de un circulo';
5
6 Definir radio como entero;
7 .....
8 Escribir 'Ingresa el radio del circulo: ';
9 leer radio;
10
11 Escribir 'el area del circulo con radio ', radio, ' es igual a: ', (3.1416 * radiot2) ;
12
13
14
15 FinProceso
16
```

An execution window titled 'PSEInt - Ejecutando proceso SIN_TITULO' is overlaid on the code. It shows the following output:

```
*** Ejecución Iniciada. ***
Este programa sirve para calcular el area de un circulo
Ingresa el radio del circulo:
> 4
el area del circulo con radio 4 es igual a: 50.2656
*** Ejecución Finalizada. ***
```

At the bottom of the IDE window, there are checkboxes for 'No cerrar esta ventana' (checked) and 'Siempre visible' (unchecked), and a 'Reiniciar' button.

La ejecución ha finalizado sin errores.

PRACTICA 6

The screenshot displays the PSeInt IDE interface. The main window shows a Pascal program titled "Proceso sin_titulo" with the following code:

```
1 Proceso sin_titulo
2
3 //Algoritmo para mostrar una cadena de datos en modo caracter. Muestra el nombre del usuario
4 //Emmanuel de Jesus Teobal Diaz
5 //Nombre(s) apellidos
6 //Apellidos Nonbre(s)
7
8 //Definir variables primero
9
10 definir nom,app, apm Como Caracter;
11
12 //Solicitar datos
13
14 Escribir 'Teclea tu(s) nombre(s); ';
15 leer nom;
16 Escribir 'Teclea tu apellido paterno: ';
17 leer app;
18 Escribir 'Teclea tu apellido materno: ';
19 leer apm;
20
21 //mostrar el nombre
22
23 Escribir 'Tu(s) nombre(s) es(son): ', nom, ' y tus apellidos son: ', app, ' ', apm ;
24
25
26 escribir 'Tus apellidos son: ', app, ' ', apm, ' y tu(s) nombre(s) son: ', nom;
27 FinProceso
28
```

An execution window titled "PSeInt - Ejecutando proceso SIN_TITULO" is overlaid on the code, showing the following output:

```
*** Ejecución Iniciada. ***
Teclea tu(s) nombre(s);
> Emmanuel de jesus
Teclea tu apellido paterno:
> Teobal
Teclea tu apellido materno:
> Diaz
Tu(s) nombre(s) es(son): Emmanuel de jesus y tus apellidos son: Teobal Diaz
Tus apellidos son: Teobal Diaz y tu(s) nombre(s) son: Emmanuel de jesus
*** Ejecución Finalizada. ***
```

At the bottom of the IDE, a status bar indicates: "La ejecución ha finalizado sin errores."

PRACTICA 7

The screenshot displays the PSeInt software interface. The main window shows a pseudocode program named "Proceso sin_titulo" with the following code:

```
1 Proceso sin_titulo
2 //Emmanuel de Jesus Teobal Diaz
3
4 //Dadas dos variables A y B, que el usuario debe teclear, se pide realizar un Pseudocodigo que intercambie
5 //los valores de ambas variables y muestre
6 // cuanto valen, al final las dos variables:
7 //entrada 2 ,9, salida 9, 2
8
9 definir a, b, aux como entero;
10
11 a←2;
12 b←9;
13
14 escribir 'El valor de A es: ', a, ' y el valor de B es: ', b;
15 aux←a;
16 a←b;
17 b←aux;
18
19 escribir 'EL valor de A, intercambiado es: ', a, ' y el valor de B es: ', b;
20
21
22 FinProceso
23
```

An execution window titled "PSeInt - Ejecutando proceso SIN_TITULO" is open, showing the output of the program:

```
*** Ejecución Iniciada. ***
El valor de A es: 2 y el valor de B es: 9
EL valor de A, intercambiado es: 9 y el valor de B es: 2
*** Ejecución Finalizada. ***
```

The execution window also includes checkboxes for "No cerrar esta ventana" (checked) and "Siempre visible" (unchecked), and a "Reiniciar" button.

On the right side of the interface, there is a "Comandos" (Commands) panel with various control flow symbols and their corresponding labels: "Hola !" (Escribir), "Dato1" (Leer), "A ← B + 1" (Asignar), "Si-Entonces" (Si-Entonces), "Según" (Según), "Mientras" (Mientras), "Repetir" (Repetir), "Para" (Para), and "SubProceso" (SubProceso).

At the bottom left, a status bar indicates: "La ejecución ha finalizado sin errores".

PRACTICA 8

The screenshot displays the PSeInt IDE interface. The main editor window shows a Pascal program titled "Proceso sin_titulo" with the following code:

```
1 Proceso sin_titulo
2 //Emmanuel de Jesus Teobal Diaz
3 //Este algoritmo permite obtener el promedio de 4 notas academicas.
4
5 Definir cal1, cal2, cal3, cal4 como entero;
6 definir prom como Real;
7
8 Escribir '**Este programa permite calcular el promedio de 4 calificaciones diferentes**';
9
10 //Proceso de lectura de datos
11
12 Escribir 'Ingrese la primera calificacion: ';
13 leer cal1;
14 Escribir 'Ingrese la segunda calificacion: ';
15 leer cal2;
16 Escribir 'Ingrese la tercera calificacion: ';
17 leer cal3;
18 Escribir 'Ingrese la cuarta calificacion: ';
19 leer cal4;
20
21 //Calculo del promedio en base a las calificaciones dadas
22 prom ← ((cal1+cal2+cal3+cal4)/4);
23 Escribir 'El promedio de la calificacion ', cal1, ', ', cal2, ', ', cal3, ' y ', cal4, ' es: ', prom;
24
25
26
27
28 FinProceso
29
```

An execution window titled "PSeInt - Ejecutando proceso SIN_TITULO" is overlaid on the code, showing the program's output:

```
*** Ejecución Iniciada. ***
**Este programa permite calcular el promedio de 4 calificaciones diferentes**
Ingrese la primera calificacion:
> 4
Ingrese la segunda calificacion:
> 7
Ingrese la tercera calificacion:
> 8
Ingrese la cuarta calificacion:
> 9
El promedio de la calificacion 4, 7, 8 y 9 es: 7
*** Ejecución Finalizada. ***
```

At the bottom of the execution window, there are checkboxes for "No cerrar esta ventana" (checked) and "Siempre visible", along with a "Reiniciar" button.

Below the IDE window, the text "a ejecución ha finalizado sin errores" is visible.

PRACTICA 9

The image shows a screenshot of a programming IDE with several tabs at the top: 'ejercicio_5.psc*', 'ejercicio_6.psc*', 'ejercicio_7.psc*', 'ejercicio_8.psc*', and 'ejercicio_9.psc* X'. The main editor window displays the following Pascal code:

```
1 Proceso sin_titulo
2     //Emmanuel de jesus Teobal Diaz
3     //Este algoritmo sirve para transformar longitudes medidas en Centimetros(Cm, a Pulgadas(In).
4     //Una pulgada es igual a 2.54 centimetros
5
6     Definir conv, long como real;
7
8     Escribir '**Este programa convierte las longitudes de centimetros(cm) a pulgadas(in)** ';
9     Escribir 'Ingrese la longitud en cm: ';
10    leer conv;
11    long ← (conv/2.54);
12
13    Escribir 'La medida de ', conv, ' centimetros, es igual a: ', long, ' pulgadas.';
14
15
16 FinProceso
17
```

Overlaid on the code is a window titled 'PSeInt - Ejecutando proceso SIN_TITULO'. The window contains the following text:

```
*** Ejecución Iniciada. ***
**Este programa convierte las longitudes de centimetros(cm) a pulgadas(in)
)**
Ingrese la longitud en cm:
> 100
La medida de 100 centimetros, es igual a: 39.3700787402 pulgadas.
*** Ejecución Finalizada. ***
```

At the bottom of the window, there are checkboxes for 'No cerrar esta ventana' (checked) and 'Siempre visible' (unchecked), and a 'Reiniciar' button.

On the right side of the IDE, there is a 'Comandos' panel with various icons for operations like 'Hola!', 'Datos', 'A ← B + 1', 'Si-Entonces', 'Mientras', 'Repetir', and 'SubProceso'.

PRACTICA 10 (TAREA)

The image shows a screenshot of a Pascal IDE with a code editor and a console window. The code editor contains the following Pascal code:

```
1 Proceso sin_titulo
2 //Emmanuel de jesus teobal diaz
3 //Este algoritmo intercambia el orden de un numero entero de 3 digitos
4
5 definir n,x,c Como Entero;
6 definir a,suma Como Real;
7
8 escribir ' Ingresa un numero: ';
9 leer n;
10 mientras n > 0 Hacer
11     x ← n mod 10;
12     escribir x;
13     n ← trunc(n/10);
14 FinMientras
15 FinProceso
16
```

The console window, titled "PSeInt - Ejecutando proceso SIN_TITULO", shows the execution output:

```
*** Ejecución Iniciada. ***
Ingresa un numero:
> 123
3
2
1
*** Ejecución Finalizada. ***
```

At the bottom of the console window, there are checkboxes for "No cerrar esta ventana" (checked) and "Siempre visible" (unchecked), along with a "Reiniciar" button.

PRACTICA 10.1 (RESUELTO EN CLASE)

The image shows a screenshot of a programming environment with a Pascal code editor and a console window. The code editor contains the following code:

```
1 Proceso sin_titulo
2 //Emmanuel de Jesus teobal diaz
3 //algoritmo para invertir los digitos de un numero de 3 cifras
4 definir number como entero;
5 escribir 'Escribe un numero de 3 digitos';
6 leer number;
7 definir unidad, decena, centena como entero;
8
9 centena ← trunc(number/100);
10 number←number%100;
11
12 decena ← trunc(number/10);
13 unidad← number% 10;
14 escribir unidad, decena, centena;
15
16
17 FinProceso
18
```

The console window, titled "PSeInt - Ejecutando proceso SIN_TITULO", shows the execution output:

```
*** Ejecución Iniciada. ***
Escribe un numero de 3 digitos
> 123
321
*** Ejecución Finalizada. ***
```

The console window also has checkboxes for "No cerrar esta ventana" (checked) and "Siempre visible" (unchecked), and a "Reiniciar" button.

On the right side of the environment, there is a "Comandos" panel with various control icons for the program execution, such as "Hola!", "Dato1", "A ← B + i", "Si-Ent", "Segu", "Mientras", and "Repeti".

PRACTICA 11

The image shows a screenshot of the PSeInt IDE. The main window displays a Pascal program for calculating the hypotenuse of a right-angled triangle. The code is as follows:

```
1 Proceso sin_titulo
2 //Emmanuel de Jesus Teobal Diaz
3 //Escriba un pseudocodigo que reciba como entrada las longitudes de los dos catetos A y B de un triangulo
4 //rectangulo y que entregue como salida el largo de la hipotenusa C del triangulo dado por el teorema de
5 //pitagoras
6
7 definir cato,cata como entero;
8 definir longi como real;
9
10 Escribir '**Programa para calcular la hipotenusa de un triangulo, dados sus catetos**';
11
12 Escribir 'Ingrese la longitud del cateto opuesto; ';
13 leer cato;
14 escribir ' Ingrese la longitud del cateto adyacente; ';
15 leer cata;
16
17 longi ← raiz((cato2)+(cata2));
18
19
20
21 Escribir 'La longitud de la hipotenusa, cuyo cateto opuesto ', cato,' y su cateto adyacente ', cata,' es: ',longi;
22
23
24 FinProceso
25
```

An execution window titled "PSeInt - Ejecutando proceso SIN_TITULO" is overlaid on the code, showing the following output:

```
*** Ejecución Iniciada. ***
**Programa para calcular la hipotenusa de un triangulo, dados sus catetos**
Ingrese la longitud del cateto opuesto;
> 7
  Ingrese la longitud del cateto adyacente;
> 8
La longitud de la hipotenusa, cuyo cateto opuesto 7 y su cateto adyacente 8 es: 10.6301458127
*** Ejecución Finalizada. ***
```

At the bottom of the execution window, there are checkboxes for "No cerrar esta ventana" (checked) and "Siempre visible" (unchecked), along with a "Reiniciar" button.

CONCLUSION

El propósito de la primera unidad fue introducirnos a la programación usando el entorno de desarrollo de PSEINT, software con el cual logramos exitosamente involucrarnos de manera superficial con la programación usando el pseudocódigo, el diagrama de flujo y el programa. A lo largo de estas primeras sesiones, aprendimos sobre algunos términos usados en la programación, tales como: Algoritmo, constante, variable, operadores, identificadores entre otros.

Elaboramos pequeños programas dedicados al estudio de estos términos, así como adaptamos estos conocimientos en pequeñas situaciones de la vida cotidiana, para entender como la programación nos sirve en cualquier ámbito, un gran ejemplo fue el programa para calcular la longitud de la hipotenusa de un triángulo rectángulo, o el programa de conversión de centímetros a pulgadas.

Personalmente disfruté de esta pequeña introducción a la programación, ya que me ayudó a entender como analizar, diseñar e implementar soluciones usando la programación.



Emmanuel de Jesus Teobal Diaz

jesus.laboet.official@gmail.com

◀ Cambiar usuario ▼ ▶

30 de 34 ▼



◀ Página 1 de 15 ▶



Entrega

Enviado para calificar

Calificado

La tarea fue enviada 2 horas 48 minutos antes

El estudiante puede editar esta entrega



 [EDJTDPRACTICASU1.pdf](#)

► Comentarios (0)

Calificación

Calificación:

Practicas

Hoja de presentación	No contien todos los datos 0 puntos	Datos incompletos 2 puntos	Completo 4 puntos	
Indice	No contiene 0 puntos	Contiene 2 puntos		

Prácticas	No contiene 0 puntos	Parcialmente 5 puntos	Mitad 10 puntos	Casi el 100% 15 puntos	Todas 20 puntos	
Conclusión	No contiene 0 puntos	Pequeña 1 puntos	Completa, falta coherencia 2 puntos	Completa y coherente 4 puntos		

Calificación actual en el libro de calificaciones

28,00

Comentarios de retroalimentación



Notificar a los estudiantes

Guardar cambios

Reiniciar



INSTITUTO TECNOLÓGICO SUPERIOR DE SAN ANDRÉS TUXTLA



EXAMEN UNIDAD 1

PLAN DE ESTUDIO: INGENIERIA INFORMATICA

ALUMNO: EMMANUEL DE JESUS TEOBAL DIAZ

DOCENTE: ENRIQUE TELONA TORRES

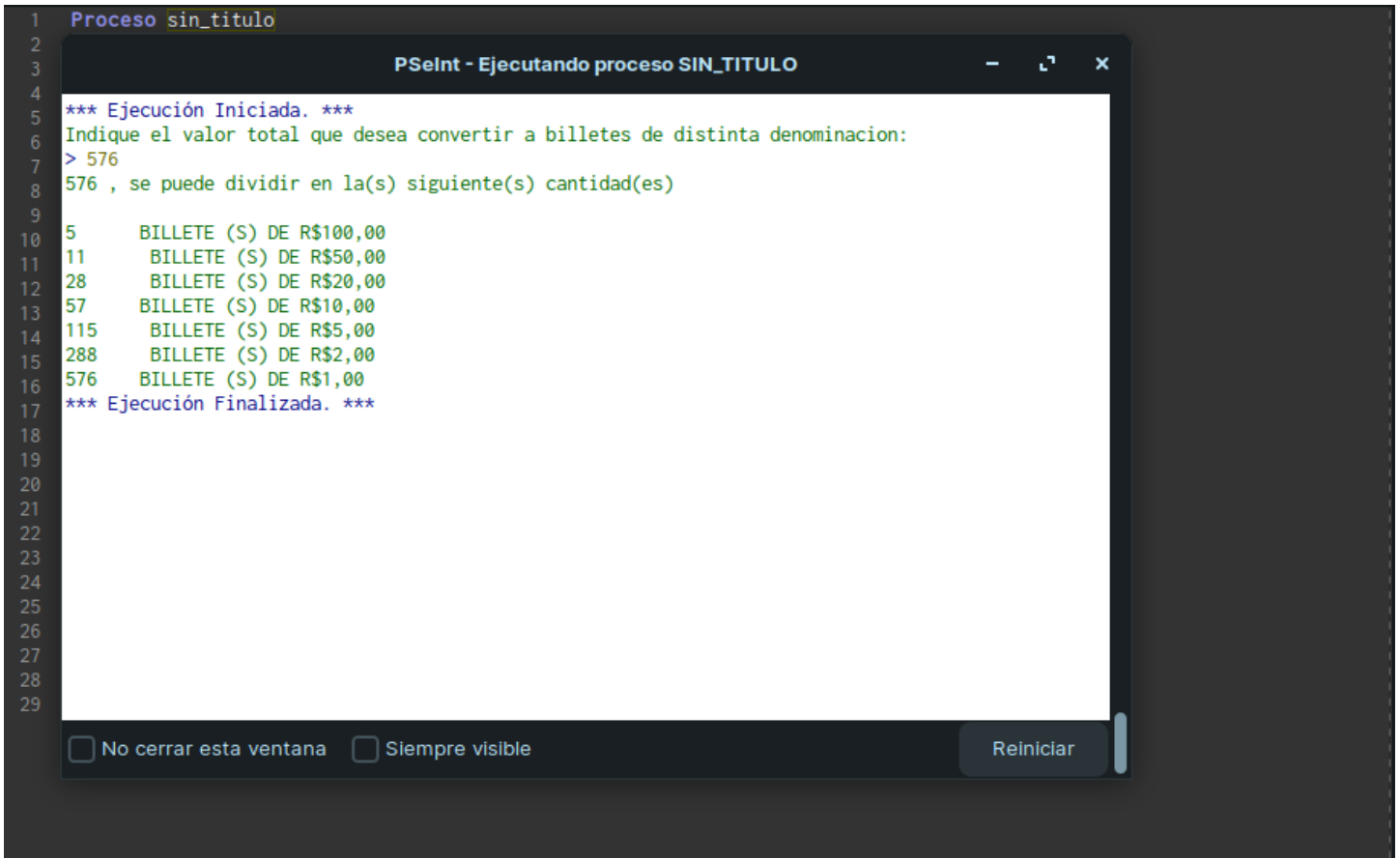
20/09/2022

PSEUDOCODIGO DEL PROGRAMA

```
1  Proceso sin_titulo
2
3  //Programa elaborado por: Osvany Fonseca y Emmanuel Teobal
4
5  Definir billete Como entero;
6  Escribir "Indique el valor total que desea convertir a billetes de distinta denominacion: ";
7  Leer billete;
8  Definir cien,cincuenta,veinte,diez,cinco,dos,uno Como Entero;
9
10 cien←trunc(billete/100);
11 cincuenta←trunc(billete/50);
12 veinte←trunc(billete/20);
13 diez←trunc(billete/10);
14 cinco←trunc(billete/5);
15 dos←trunc(billete/2);
16 uno←(billete/1);
17
18 Escribir billete, ' , se puede dividir en la(s) siguiente(s) cantidad(es)';
19 Escribir '';
20 Escribir cien,'      BILLETE (S) DE R$100,00';
21 Escribir cincuenta,'    BILLETE (S) DE R$50,00' ;
22 Escribir veinte,'      BILLETE (S) DE R$20,00';
23 Escribir diez,'        BILLETE (S) DE R$10,00';
24 Escribir cinco,'       BILLETE (S) DE R$5,00';
25 Escribir dos,'         BILLETE (S) DE R$2,00';
26 Escribir uno,'        BILLETE (S) DE R$1,00';
27
28 FinProceso
29
```

IMPRESIÓN DE RESULTADOS

```
1 Proceso sin_titulo
2
3
4
5 *** Ejecución Iniciada. ***
6 Indique el valor total que desea convertir a billetes de distinta denominacion:
7 > 576
8 576 , se puede dividir en la(s) siguiente(s) cantidad(es)
9
10 5      BILLETE (S) DE R$100,00
11 11     BILLETE (S) DE R$50,00
12 28     BILLETE (S) DE R$20,00
13 57     BILLETE (S) DE R$10,00
14 115    BILLETE (S) DE R$5,00
15 288    BILLETE (S) DE R$2,00
16 576    BILLETE (S) DE R$1,00
17 *** Ejecución Finalizada. ***
18
19
20
21
22
23
24
25
26
27
28
29
```



(INTRODUJENDO 576)

```
1 Proceso sin_titulo
2
3
4
5 *** Ejecución Iniciada. ***
6 Indique el valor total que desea convertir a billetes de distinta denominacion:
7 > 11257
8 11257 , se puede dividir en la(s) siguiente(s) cantidad(es)
9
10 112      BILLETE (S) DE R$100,00
11 225      BILLETE (S) DE R$50,00
12 562      BILLETE (S) DE R$20,00
13 1125     BILLETE (S) DE R$10,00
14 2251     BILLETE (S) DE R$5,00
15 5628     BILLETE (S) DE R$2,00
16 11257    BILLETE (S) DE R$1,00
17 *** Ejecución Finalizada. ***
18
19
20
21
22
23
24
25
26
27
28
29
```

No cerrar esta ventana Siempre visible Reiniciar

(INTRODUJENDO 11257)

```
1  Proceso sin_titulo
2
3
4
5  *** Ejecución Iniciada. ***
6  Indique el valor total que desea convertir a billetes de distinta denominacion:
7  > 503
8  503 , se puede dividir en la(s) siguiente(s) cantidad(es)
9
10 5      BILLETE (S) DE R$100,00
11 10     BILLETE (S) DE R$50,00
12 25     BILLETE (S) DE R$20,00
13 50     BILLETE (S) DE R$10,00
14 100    BILLETE (S) DE R$5,00
15 251    BILLETE (S) DE R$2,00
16 503    BILLETE (S) DE R$1,00
17 *** Ejecución Finalizada. ***
18
19
20
21
22
23
24
25
26
27
28
29
```

PSelnt - Ejecutando proceso SIN_TITULO

No cerrar esta ventana Siempre visible Reiniciar

(INTRODUJENDO 503)

```
1 Proceso sin_titulo
2
3
4
5 *** Ejecución Iniciada. ***
6 Indique el valor total que desea convertir a billetes de distinta denominacion:
7 > 1
8 1 , se puede dividir en la(s) siguiente(s) cantidad(es)
9
10 0 BILLETE (S) DE R$100,00
11 0 BILLETE (S) DE R$50,00
12 0 BILLETE (S) DE R$20,00
13 0 BILLETE (S) DE R$10,00
14 0 BILLETE (S) DE R$5,00
15 0 BILLETE (S) DE R$2,00
16 1 BILLETE (S) DE R$1,00
17 *** Ejecución Finalizada. ***
18
19
20
21
22
23
24
25
26
27
28
29
```

No cerrar esta ventana Siempre visible Reiniciar

(INTRODUJENDO 1)


```
1 Proceso sin_titulo
2
3 PSeInt - Ejecutando proceso SIN_TITULO
4
5 *** Ejecución Iniciada. ***
6 Indique el valor total que desea convertir a billetes de distinta denominacion:
7 > 10
8 10 , se puede dividir en la(s) siguiente(s) cantidad(es)
9
10 0 BILLETE (S) DE R$100,00
11 0 BILLETE (S) DE R$50,00
12 0 BILLETE (S) DE R$20,00
13 1 BILLETE (S) DE R$10,00
14 2 BILLETE (S) DE R$5,00
15 5 BILLETE (S) DE R$2,00
16 10 BILLETE (S) DE R$1,00
17 *** Ejecución Finalizada. ***
18
19
20
21
22
23
24
25
26
27
28
29
```

No cerrar esta ventana Siempre visible Reiniciar

(INTRODUJENDO 10)



Emmanuel de Jesus Teobal Diaz

jesus.laboet.official@gmail.com

◀ Cambiar usuario ▼ ▶

30 de 34 ▼



◀ Página 1 de 7 ▶



Entrega

Enviado para calificar

Calificado

La tarea fue enviada 39 minutos 19 segundos antes

El estudiante puede editar esta entrega



 [EDJTDExaU1.pdf](#)

► Comentarios (0)

Calificación

Calificación:

Practicas

PDF	Otro formato 0 puntos	Con formato 1 puntos	
Reporte con capturas	No envió 0 puntos	Si envió 4 puntos	

Funcionamiento de código	No funciona	Soluciona 1 prueba	Soluciona 2 pruebas	Soluciona 3 pruebas	Soluciona 4 pruebas	Soluciona 5 pruebas	
	0 puntos	7 puntos	14 puntos	21 puntos	28 puntos	35 puntos	

Calificación actual en el libro de calificaciones

26,00

Comentarios de retroalimentación



Notificar a los estudiantes

Guardar cambios

Reiniciar