LÓGICA
Y DIAGRAMACIÓN

Instituto Universitario de Tecnología para la Informática

CARRERA: ANÁLISIS DE SISTEMAS
SEMESTRE 1ro

Introducción a la Lógica

¿Qué es la Lógica?

La lógica es la ciencia que estudia los principios y métodos para distinguir el razonamiento correcto del incorrecto. Es una herramienta fundamental en

diversas disciplinas, desde las matemáticas y la filosofía hasta la informática y la inteligencia artificial.

¿Por qué Estudiar Lógica?

- **Pensamiento crítico:** Desarrolla la capacidad de analizar argumentos, identificar falacias y evaluar la validez de las conclusiones.
- **Resolución de problemas:** Te enseña a abordar problemas de manera sistemática y encontrar soluciones efectivas.
- **Comunicación clara:** Mejora tu habilidad para expresar ideas de forma precisa y concisa.
- **Toma de decisiones:** Te ayuda a evaluar las opciones y tomar decisiones informadas.

Gustavo A. Jiménez

Licenciado en Educación mención Informática
T.S.U. en Análisis de Sistemas.

Una publicación de

Instituto Universitario de Tecnología para la Informática

Introducción

La Lógica es una disciplina fundamental en el estudio de Análisis de Sistemas, ya que proporciona las herramientas necesarias para la resolución de problemas, el razonamiento estructurado y la toma de decisiones informadas. En el contexto de la informática y la programación, la lógica no solo es un componente esencial para entender cómo funcionan los sistemas computacionales, sino que también es la base para diseñar y desarrollar algoritmos eficientes, estructuras de datos, y modelos de programación que soportan el funcionamiento de sistemas complejos.

A lo largo de este ebook, exploraremos los conceptos fundamentales de la lógica, tales como la lógica proposicional y la lógica de predicados, los cuales son cruciales para comprender los procesos de inferencia y deducción que subyacen en la creación y análisis de sistemas. También se abordarán temas clave como los operadores lógicos, las tablas de verdad, y las reglas de inferencia, proporcionando una base teórica sólida para los futuros desafíos en el campo del análisis y desarrollo de sistemas.

El objetivo de este ebook es ofrecer a los estudiantes de Análisis de Sistemas un recurso completo y accesible que los guíe desde los conceptos más básicos hasta aplicaciones más avanzadas de la lógica en el campo de la informática. A través de ejemplos prácticos y ejercicios aplicados, se busca que los lectores desarrollen habilidades críticas para la formulación de soluciones algorítmicas, el pensamiento lógico-matemático y el diseño de sistemas eficientes.

Plan de Estudio

Unidad I

Definición de Lenguaje, Definición de términos básicos, Lógica, Pseudocódigo, Algoritmo, Constantes, Datos, Variables, Contadores, Tabla de la Verdad.

Unidad II

Operadores matemáticos, relacionales, lógicos, algoritmos unidireccionales, secuenciales, Condicionales SI, SINO, prácticas con Pseudocodigo.

Unidad III

Estructuras repetitivas, ciclo PARA, MIENTRAS, HACER PARA, integración al Software Educativo Pseint.

Unidad IV

Prácticas de operadores lógicos, matemáticos, relacionales junto con estructuras repetitivas y condicionales en Software Educativo PSEINT.

Unidad V

Trabajo con objetos unidimensionales Vectores y Bidimensionales Matrices en Software Educativo PSEINT

ÍNDICE

Lcdo. Gustavo Jiménez

Unidad I

Conceptos básicos de la lógica proposicional y de predicados.

El lenguaje es una herramienta fundamental para la humanidad, se desarrolla desde la infancia y que a través del tiempo ha ido manifestándose de variadas formas, con el fin de transmitir el conocimiento de una generación a otra.

Es una capacidad que diferencia a los humanos de los animales. Al hablar sobre el lenguaje, se puede encontrar que está catalogada en varios tipos, cada uno con una función determinada, pero que de alguna forma se entrelazan complementan entre sí.

El lenguaje:

En términos generales, es un sistema de comunicación estructurado que permite la transmisión de ideas, información, emociones o instrucciones a través de signos (como palabras, símbolos o sonidos) que son entendibles para quienes comparten ese sistema.

Se clasifican según:

1-Lenguaje Informativo:

El lenguaje informativo es el propio de los medios de comunicación, tanto escritos como audiovisuales, pero también podría entenderse por lenguaje informativo aquel que utilizan entidades u organismos cuando tienen intención de comunicar un hecho aséptico. La función informativa, o cognitiva, se ve cuando decimos algo para expresar ideas o transmitir información. Algunos

ejemplos son: Tengo dos hermanos; Hay un bolígrafo en la mesa; Vivo en la casa rosada a la izquierda; México está compuesto de 31 estados y la capital federal.

2-Lenguaje Comunicativo:

Es una capacidad racional que tienen todos los seres humanos, por lo que es universal. Es la base de comunicación entre los individuos. Resulta clave en las relaciones interpersonales. Se vale de las diferentes lenguas para codificar un mensaje y luego transmitirlo. Es un tipo de lenguaje que requiere el uso de palabras para establecer la comunicación. Ejemplo: una llamada telefónica, el dictado de una clase, etc.

3-Lenguaje Directivo:

Es usado para causar, impedir acciones, órdenes o peticiones. No es ni verdadero ni falso. El lenguaje directo cumple la función de dar una orden o impedir una acción. La diferencia de una orden es que en el lenguaje escrito puede acompañarse de una expresión como «por favor» y en el lenguaje verbal con los cambios adecuados de voz. Algunos ejemplos incluyen: ¡Ven aquí!; ¡Dímelo!; ¿Puedes pasarme el pan?; Te sugiero que te vayas…; ¿No quieres venir conmigo?

4-Lenguaje Formal:

El lenguaje formal emplea un vocabulario sofisticado y evita la jerga o los coloquialismos. Es el que utilizamos en situaciones más serias y formales. Planificación, elección y uso de los recursos lingüísticos adecuados. Por ejemplo: en una entrevista de trabajo, en una cita con tu médico o cuando conoces a una persona por primera vez.

5-Lenguaje Natural:

Es una variedad lingüística o forma de lenguaje humano generada espontáneamente en un grupo de hablantes con propósito de comunicarse, a diferencia de otras lenguas, como puedan ser una lenguaje construida, los lenguaje de programación o los lenguajes formales usados en el estudio de la lógica formal, especialmente la lógica matemática. Para servir a su propósito de comunicación, una lengua natural ha de disponer de una gramática (sintaxis, morfología, etc.). Por ejemplo, el castellano, el catalán, el vasco o el gallego, en España, y cualquier otro idioma que se hable en alguna parte del mundo entero.

6-Lenguaje Informal:

Es el utilizado del lenguaje en un entorno informal y familiar. Se utiliza en la conversación natural y cotidiana. Lo utilizamos con personas cercanas, con las que nos sentimos relajados a la hora de comunicarnos: familiares, amigos, compañeros de clase, etc. Ejemplos, ¿Bueno? ¿Quién habla?, Manito, me caes muy bien, etc.

7-Lenguaje de Señas:

La lengua de señas es la lengua natural de las personas sordas. A diferencia del lenguaje oral, la lengua de señas se basa en las expresiones faciales y en diversos movimientos de las manos, los brazos y el cuerpo. La LSM, como todo lenguaje, posee su propia gramática. La lengua de señas es la lengua natural de las personas sordas. A diferencia del lenguaje oral, la lengua de señas se basa en las expresiones faciales y en diversos movimientos de las manos, los brazos y el cuerpo. La LSM, como todo lenguaje, posee su propia gramática.

8-Lenguaje Expresivo:

Es lo que una persona trata de transmitir hacia otra por medio de gestos, hablado, escrito o por señas. El lenguaje expresivo puede ser tanto de forma verbal como no verbal. Algunos ejemplos de lenguaje expresivo son: señalar con el dedo, llorar, gritar, balbucear, nombrar objetos, solicitar objetos, expresar necesidades, responder a preguntas, compartir situaciones que suceden en su entorno, entre otras.

9- El lenguaje en informática:

El lenguaje cumple una función similar pero con el objetivo de comunicarse entre humanos y máquinas. Es un conjunto de reglas y símbolos (también llamado sintaxis) que permite que los programadores den instrucciones a las computadoras para que ejecuten tareas específicas. Los lenguajes de programación, por ejemplo, son diseñados para que las personas puedan escribir comandos que una máquina pueda traducir y ejecutar.

El concepto de lenguaje en informática se basa en dos componentes principales:

Sintaxis: Las reglas que definen cómo se deben estructurar las instrucciones.
Semántica: El significado de esas instrucciones; es decir, lo que cada comando o sentencia logra hacer en el sistema.

Es por ello, que el lenguaje juega un papel crucial en la lógica de predicados, ya que proporciona la estructura y las herramientas necesarias para expresar y manipular proposiciones complejas de manera precisa y formal. Aquí hay una explicación detallada sobre cómo el lenguaje influye en la lógica de predicados:

Sintaxis del Lenguaje de Predicados

La sintaxis es el conjunto de reglas que define cómo se pueden formar expresiones válidas en un lenguaje. En la lógica de predicados, la sintaxis especifica cómo se construyen las proposiciones a partir de los elementos básicos del lenguaje:

Variables: Representan objetos en el dominio de discurso (por ejemplo, x, y, z).

Constantes: Representan objetos específicos en el dominio (por ejemplo, a, b, c).

Predicados: Funciones que describen propiedades de objetos o relaciones entre ellos (por ejemplo, P(x), Q(x, y)).

Cuantificadores: Los cuantificadores universales ($\forall$) y existencial ($\exists$) permiten generalizar proposiciones sobre conjuntos de objetos.

Conectivos Lógicos: Incluyen y ($\wedge$), o ($\vee$), no ($\neg$), implica ($\rightarrow$), y si y solo si ($\leftrightarrow$).

Semántica del Lenguaje de Predicados

La semántica se refiere al significado de las expresiones en el lenguaje. En la lógica de predicados, la semántica se encarga de interpretar las proposiciones y determinar su verdad o falsedad:

Interpretación: Asignación de significados a los símbolos del lenguaje (por ejemplo, asignar un objeto específico del dominio a una constante, o una relación específica a un predicado).

Valor de Verdad: Determinación de si una proposición es verdadera o falsa bajo una interpretación dada.

Expresividad del Lenguaje

La lógica de predicados permite expresar proposiciones que no pueden ser formuladas en lógica proposicional, debido a la inclusión de variables y cuantificadores. Por ejemplo:

En lógica proposicional, solo podemos expresar proposiciones simples y sus combinaciones, como "P ∧ Q" o "¬R".

Expresividad del Lenguaje

La lógica de predicados permite expresar proposiciones que no pueden ser formuladas en lógica proposicional, debido a la inclusión de variables y cuantificadores. Por ejemplo:

En lógica proposicional, solo podemos expresar proposiciones simples y sus combinaciones, como "P ∧ Q" o "¬R".

En lógica de predicados, podemos expresar proposiciones más complejas que involucran relaciones entre objetos, como "Todos los estudiantes en la clase han aprobado el examen" (∀x (Estudiante(x) → Aprobado(x))).

- **¿Qué es un enunciado?**

Un enunciado es un grupo de palabras ordenadas que tiene sentido completo. Por ejemplo: Eva será una gran científica. Hay dos clases de enunciados: oracionales y no oracionales. Los enunciados oracionales u oraciones contienen al menos una forma verbal. Se puede decir que los enunciados son frases o expresiones habladas o escritas que son expresadas para decir algo, estas pueden expresar un deseo o una verdad que pueden ser verdaderas o falsas o también ser expresadas de forma exclamativa o como una pregunta.

Por ejemplo: "Perro come hueso el" NO sería un enunciado porque no está ordenado ni tiene sentido completo. "El perro come el hueso" SI sería un enunciado, porque está ordenado y tiene sentido completo.

- **¿Qué es una oración?**

Una oración es una unidad de sentido que presenta autonomía sintáctica, es decir, que puede funcionar de manera independiente. Por ejemplo: El juez dictó su sentencia. Las oraciones son las unidades mínimas del discurso que permiten comunicar una idea completa. Una oración es una unidad de sentido compuesta por diferentes palabras ordenadas que expresan una idea o mensaje. Es el fragmento más básico del discurso y su objetivo es comunicar, por lo que siempre debe tener sentido (dentro de un contexto) y coherencia. Por ejemplo: Los niños visitarán la muestra esta tarde.

- **¿Qué es el razonamiento?**

Se entiende por razonamiento al proceso mental a través del cual una persona utiliza la información disponible para llegar de manera lógica a una conclusión, solución, o idea. El razonamiento se basa en la capacidad de procesar

información, conectar ideas y utilizar la lógica para llegar a conclusiones válidas. Un razonamiento puede darse en un enunciado formal, como pasa en el lenguaje propio de la lógica, o en un enunciado informal.

Un ejemplo de enunciado informal es el razonamiento argumentativo. Los razonamientos argumentativos son toda actividad mental que se corresponda con la actividad lingüística de argumentar. Ejemplo, Para salir de viaje con mis amigos necesito tener suficiente dinero, si ahorro todos los meses parte de mi sueldo, entonces podré viajar con ellos.

Términos Básicos en Lógica

1. **Proposición:**
    a. Definición: Una proposición es una declaración que puede ser verdadera o falsa, pero no ambas al mismo tiempo. Ejemplo: "El cielo es azul" es una proposición.

2. **Conectivos Lógicos:**
    a. Y (∧): Conjunción. La proposición "P ∧ Q" es verdadera si y solo si ambas P y Q son verdaderas.
    b. O (∨): Disyunción. La proposición "P ∨ Q" es verdadera si al menos una de P o Q es verdadera.
    c. No (¬): Negación. La proposición "¬P" es verdadera si P es falsa.
    d. Implicación (→): Si... entonces. La proposición "P → Q" es verdadera si P es falsa o Q es verdadera.
    e. Doble implicación (↔):** Si y solo si. La proposición "P ↔ Q" es verdadera si P y Q tienen el mismo valor de verdad.

3. **Lógica Proposicional:**
    a. Definición: Rama de la lógica que estudia proposiciones y sus combinaciones mediante conectivos lógicos. No considera la estructura interna de las proposiciones.
4. **Lógica de Predicados (Lógica de Primer Orden):**
    a. Definición: Extensión de la lógica proposicional que incluye variables, cuantificadores y predicados para expresar proposiciones más complejas.
5. **Predicado:**
    a. Definición: Función que toma una o más variables y devuelve un valor de verdad. Ejemplo: "EsMayor(x, y)" puede ser un predicado que representa "x es mayor que y".
6. **Variable:**
    a. Variables: En lógica de predicados, una variable es un símbolo que puede representar cualquier elemento de un dominio específico. Las variables se utilizan para generalizar proposiciones y argumentos, permitiendo la formulación de enunciados que se aplican a múltiples casos, se define con un símbolo que representa un objeto en el dominio de discurso. Ejemplo: x, y, z.
7. **Constante:**
    a. En contraste, una constante es un símbolo que representa un elemento específico y fijo del dominio. Las constantes se usan para referirse a objetos particulares dentro de un argumento, se definen con un simbolo que representa un objeto específico en el dominio de discurso. Ejemplo: a, b, c.

8. **Fórmula:**
    a. Definición: Combinación de átomos y conectivos lógicos que constituye una proposición en lógica proposicional o lógica de predicados. Ejemplo: $P(x) \wedge Q(x, y)$.
9. **Interpretación:**
    a. Definición: Asignación de significados a los símbolos de un lenguaje lógico. En lógica de predicados, esto incluye asignar objetos del dominio a variables y valores de verdad a predicados.

Tabla de la verdad.

Las tablas de verdad es una estrategia de la lógica simple que permite establecer la validez de varias propuestas en cuanto a cualquier situación, es decir, permite determinar las condiciones necesarias para que sea verdadero un enunciado propuesto, permitiendo clasificarlos en tautológicos (resultan verdaderos durante cualquier situación) contradictorias (son enunciados falsos en la mayoría de los casos) o contingentes (enunciados que no pueden ser tantos verdaderos como falsos no existen tendencia a un solo sentido).

Funciones de las tablas de verdad.

- Las tablas de verdad lógicas permiten el análisis de cualquier fórmula para encontrar los valores que la hagan verdad.
- Determinan si una fórmula es satisfactoria, así como la validez de un razonamiento.

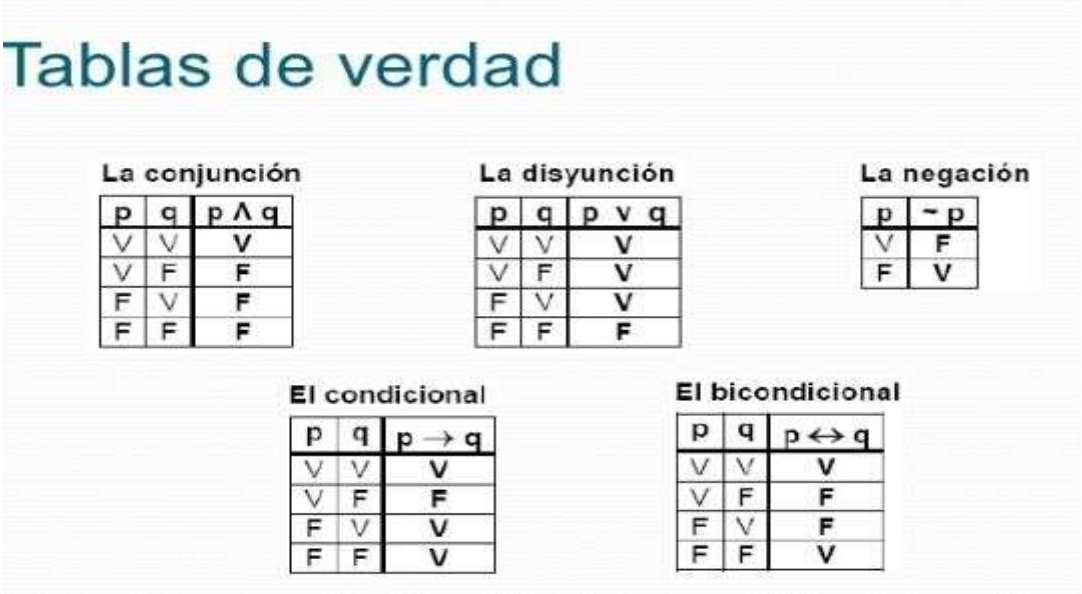


Imagen N°1. Tabla de la Verdad
Fuente: **Gustavo Jiménez 2024**

Definición de Pseudocódigo

El pseudocódigo es una representación informal de un algoritmo que utiliza una mezcla de lenguaje natural y convenciones de programación para describir los pasos que debe seguir un algoritmo. No sigue la sintaxis de ningún lenguaje de programación específico, pero es lo suficientemente estructurado como para ser fácilmente traducido a código real.

Características del Pseudocódigo:

1. Legibilidad: Está diseñado para ser fácilmente comprendido por humanos, independientemente de su conocimiento de lenguajes de programación específicos.

2. Flexibilidad: No está limitado por las reglas estrictas de la sintaxis de los lenguajes de programación, lo que permite enfocarse en la lógica del algoritmo.

3. Estructura: Utiliza estructuras comunes de control de flujo como bucles (for, while), condicionales (if, else), y procedimientos (funciones, subrutinas).

Ejemplo de Pseudocódigo:

1. **Algoritmo para encontrar el mayor de dos números**

```
Inicio
    Leer numero1, numero2
    Si numero1 > numero2 Entonces
        Escribir "El mayor es ", numero1
    Sino
        Escribir "El mayor es ", numero2
    FinSi
Fin
```

2. **Algoritmo para calcular la suma de los primeros n números enteros.**

```
 Inicio
    Leer n
    suma <- 0
    Para i desde 1 hasta n Hacer
        suma <- suma + i
    FinPara
    Escribir "La suma es ", suma
```

```
Fin
```

3. **Algoritmo para verificar si un número es par o impar.**

```
Inicio
    Leer numero
    Si numero MOD 2 = 0 Entonces
        Escribir "El número es par"
    Sino
        Escribir "El número es impar"
    FinSi
Fin
```

4. **Ejemplo de pseudocódigo: Imprimir los números del 1 al 10**

```
INICIO
    PARA i = 1 HASTA 10 HACER
        Mostrar i
    FIN_PARA
FIN
```

5. Ejemplo de pseudocódigo: Calcular el factorial de un número

```
INICIO
    Leer numero
```

```
    factorial = 1
    PARA i = 1 HASTA numero HACER
        factorial = factorial * i
    FIN_PARA
    Mostrar "El factorial es:", factorial
FIN
```

6. Ejemplo de pseudocódigo: Buscar un valor en una lista

```
INICIO
    Leer valorBuscado
    encontrado = FALSO
    PARA cada elemento EN lista HACER
        SI elemento == valorBuscado ENTONCES
            encontrado = VERDADERO
            Mostrar "Valor encontrado"
            SALIR_DEL_BUCLE
        FIN_SI
    FIN_PARA
    SI encontrado == FALSO ENTONCES
        Mostrar "Valor no encontrado"
    FIN_SI
FIN
```

Definición de Algoritmo

Un algoritmo es un conjunto finito y ordenado de instrucciones o pasos que se siguen para resolver un problema específico o realizar una tarea determinada.

Los algoritmos están diseñados para ser implementados en un lenguaje de programación y ejecutados por una computadora, aunque también pueden describirse de manera abstracta para ser entendidos por humanos.

Características de un Algoritmo:

1. Finitud: Un algoritmo debe terminar después de un número finito de pasos.
2. Precisión: Cada paso del algoritmo debe estar claramente definido y ser no ambiguo.
3. Entrada: Un algoritmo debe aceptar cero o más entradas.
4. Salida: Un algoritmo debe producir al menos una salida.
5. Efectividad: Las operaciones a realizar deben ser suficientemente básicas como para ser realizadas en un tiempo finito con precisión.

Ejemplo de un Algoritmo:

1. Algoritmo para calcular el promedio de una lista de números

```
Inicio
    Leer n
    suma <- 0
    Para i desde 1 hasta n Hacer
```

```
    Leer numero
    suma <- suma + numero
  FinPara
  promedio <- suma / n
  Escribir "El promedio es ", promedio
Fin
```

2. Algoritmo para verificar si un número es primo

```
Inicio
  Leer numero
  esPrimo <- Verdadero
  Si numero <= 1 Entonces
    esPrimo <- Falso
  Sino
    Para i desde 2 hasta numero-1 Hacer
      Si numero MOD i = 0 Entonces
        esPrimo <- Falso
        Salir
      FinSi
    FinPara
  FinSi
  Si esPrimo Entonces
```

```
        Escribir "El número es primo"
    Sino
        Escribir "El número no es primo"
    FinSi
Fin
```

3. Algoritmo para encontrar el máximo en un arreglo

```
Inicio
    Leer n
    Leer arreglo de tamaño n
    maximo <- arreglo[1]
    Para i desde 2 hasta n Hacer
        Si arreglo[i] > maximo Entonces
            maximo <- arreglo[i]
        FinSi
    FinPara
    Escribir "El máximo es ", maximo
Fin
```

4. Algoritmo para ordenar un arreglo en orden ascendente (Método de selección)

```
Inicio
```

```
    Leer n
    Leer arreglo de tamaño n
    Para i desde 1 hasta n-1 Hacer
        minIndex <- i
        Para j desde i+1 hasta n Hacer
            Si arreglo[j] < arreglo[minIndex] Entonces
                minIndex <- j
            FinSi
        FinPara
        Si minIndex ≠ i Entonces
            Intercambiar arreglo[i] con arreglo[minIndex]
        FinSi
    FinPara
    Escribir "Arreglo ordenado: ", arreglo
Fin
```

5. Algoritmo para encontrar la cantidad de veces que un elemento aparece en una lista

```
Inicio
    Leer n
    Leer lista de tamaño n
    Leer valorBuscado
```

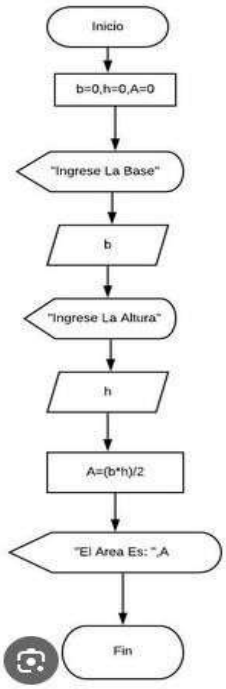
```
    contador <- 0
    Para i desde 1 hasta n Hacer
        Si lista[i] = valorBuscado Entonces
            contador <- contador + 1
        FinSi
    FinPara
    Escribir "El valor ", valorBuscado, " aparece ", contador, " veces."
Fin
```

Diferencias y Relación Entre Algoritmo y Pseudocódigo

-Formalidad: Un algoritmo es una secuencia abstracta de pasos claramente definidos para resolver un problema, mientras que el pseudocódigo es una herramienta utilizada para describir estos pasos de manera más legible para humanos.

- **Implementación:** Los algoritmos se pueden implementar en cualquier lenguaje de programación, mientras que el pseudocódigo es independiente del lenguaje y sirve como intermediario entre la idea abstracta y la implementación concreta.

- **Propósito:** El propósito del pseudocódigo es facilitar la comprensión y la comunicación del algoritmo antes de su implementación en un lenguaje de programación.



Algoritmo

-Inicio
-Leer La Base Del Triangulo
-Almacenar en una variable el valor ingresado para la base
-Leer La Altura Del Triangulo
-Almacenar en otra variable el valor ingresado para la altura
-Calcular el area del triangulo con la formula A=(b*h)/2
-Presentar en Pantalla El Area Del Triangulo
-Fin

Pseudocódigo

-Inicio
-Declarar variables b, h, A
-Pedir La Base Del Triangulo
-Guardar la base en b
-Leer La Altura Del Triangulo
-Guardar la altura en h
-Calcular el area del triangulo
A=(b*h)/2
-Escribir en Pantalla "El Area Es ",A
-Fin

Imagen N°2. Ejemplo de Diagrama de Flujo
Fuente: **Gustavo Jiménez 2024**

Diagramas de Flujo

Un diagrama de flujo es una representación gráfica de un proceso, sistema o algoritmo. Utiliza símbolos estándar y flechas para mostrar los pasos secuenciales, las decisiones, las entradas y las salidas del proceso. Los diagramas de flujo son herramientas visuales que ayudan a entender, documentar y comunicar cómo funciona un proceso o algoritmo.

Símbolos en los Diagramas de Flujo

Imagen N°3. Símbolos del Diagrama de Flujo
Fuente: **http://contenidos.sucerman.com/ 2024 Ejemplo de Diagrama de Flujo**

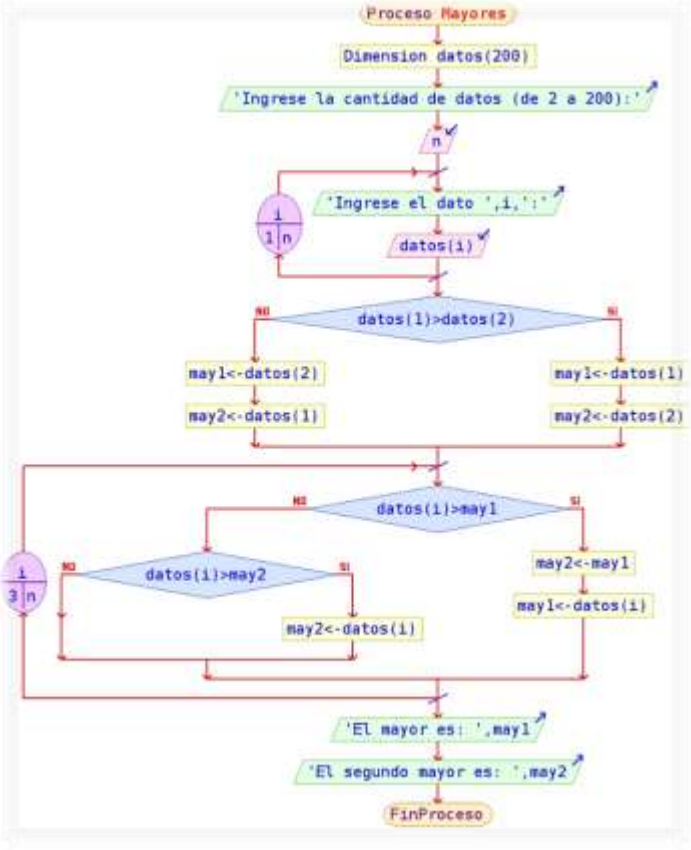


Imagen N°4. Ejemplo de Diagrama de Flujo
Fuente: **Gustavo Jiménez 2024**

Pasos para Crear un Diagrama de Flujo

1. **Definir el Proceso**: Determina el proceso o algoritmo que necesitas representar.

2. **Identificar los Pasos:** Enumera todos los pasos, decisiones, entradas y salidas involucradas en el proceso.

3. **Seleccionar los Símbolos:** Utiliza los símbolos estándar para representar cada tipo de acción o decisión.

4. **Dibujar el Diagrama:** Comienza con el símbolo de inicio, luego añade los símbolos correspondientes a cada paso en la secuencia correcta, conectándolos con flechas.

5. **Verificar y Refinar:** Revisa el diagrama para asegurarte de que sea claro y preciso. Realiza ajustes si es necesario.

Ventajas de los Diagramas de Flujo

1. **Comunicación Efectiva**: Facilitan la comunicación entre diferentes partes interesadas al proporcionar una representación visual clara del proceso.

2. **Documentación:** Sirven como documentación formal de procesos y algoritmos, útil para la capacitación y el mantenimiento.

3. **Identificación de Problemas:** Ayudan a identificar ineficiencias, cuellos de botella y áreas de mejora en un proceso.

4. **Facilidad de Análisis:** Permiten un análisis más fácil y rápido de procesos complejos, haciendo visibles las relaciones y secuencias entre los pasos.

Aplicaciones de los Diagramas de Flujo

- **Programación:** Documentación y diseño de algoritmos antes de la codificación.
- **Negocios:** Mapeo de procesos de negocio para análisis y mejora de la eficiencia.
- **Ingeniería:** Modelado de procesos de ingeniería y sistemas de producción.
- Educación: Enseñanza de conceptos y procedimientos complejos de manera visual y accesible.

Lcdo. Gustavo Jiménez

Unidad II

Operadores Matemáticos Básicos

1. Suma (+):

-Definición: Operador binario que agrega dos números.

-Ejemplo: (a + b)

- Si (a = 3) y(b = 5), entonces (3 + 5 = 8).

2. Resta (−):

- Definición: Operador binario que encuentra la diferencia entre dos números.

- Ejemplo: (a - b)

- Si (a = 10) y (b = 4), entonces (10 - 4 = 6).

3. Multiplicación (×, *):

- Definición: Operador binario que encuentra el producto de dos números.

- Ejemplo: (a * b)

- Si (a = 4) y (b = 7), entonces (4 * 7 = 28).

4. División (÷):

-Definición: Operador binario que encuentra el cociente de dos números.

- Ejemplo: (a ÷ b) o (a / b)

- Si (a = 20) y (b = 5), entonces (20 ÷ 5 = 4).

Operadores Matemáticos Avanzados

1. Potenciación (^):

- Definición: Operador binario que eleva un número a la potencia de otro.
- *Ejemplo: (a^b) o (a ^ b)
- Si (a = 2) y (b = 3), entonces (2^3 = 8).

2. Módulo (%):

- Definición: Operador binario que encuentra el residuo de la división de un número por otro.
- Ejemplo: (a % b %)
- Si (a = 10) y (b = 3), entonces (10 % 3 = 1).

Operadores Relacionales en Lógica

1. Igualdad (=):

- Definición: Operador binario que verifica si dos expresiones son iguales.
- Ejemplo: (a = b)
- Si (a = 5) y (b = 5), entonces (5 = 5) es verdadero.

2. Desigualdad (≠, !=):

- Definición: Operador binario que verifica si dos expresiones no son iguales.
- Ejemplo: (a ≠ b) o (a != b)
- Si \(a = 5 \) y \(b = 4 \), entonces \(5 ≠ 4 \) es verdadero.
-

3. Menor que (<):

- Definición: Operador binario que verifica si el primer número es menor que el segundo.
- Ejemplo: \(a < b \)
- Si \(a = 3 \) y \(b = 5 \), entonces \(3 < 5 \) es verdadero.

4. Mayor que (>):

- Definición: Operador binario que verifica si el primer número es mayor que el segundo.
- Ejemplo: (a > b)
- Si (a = 7) y (b = 2), entonces (7 > 2) es verdadero.

5. Menor o igual que (≤, <=):

- Definición: Operador binario que verifica si el primer número es menor o igual que el segundo.
- Ejemplo: (a ≤ b) o (a <= b)
- Si (a = 4) y (b = 4), entonces (4 ≤ 4) es verdadero.

6. Mayor o igual que (≥, >=):

- Definición: Operador binario que verifica si el primer número es mayor o igual que el segundo.
- Ejemplo: (a ≥ b) o (a >= b)
- Si (a = 6) y (b = 5), entonces (6 ≥ 5) es verdadero.

Lcdo. Gustavo Jiménez

Operadores Relacionales:

Los operadores relacionales son fundamentales en la lógica, especialmente en la lógica de predicados, ya que nos permiten expresar relaciones entre términos o variables. Estos operadores comparan dos términos y devuelven un valor de verdad (verdadero o falso) en función de dicha comparación.

Principales Operadores Relacionales

Igualdad (=): Verifica si dos términos son iguales.

Ejemplo: x=yx=y es verdadero si xx y yy son iguales.

Desigualdad (≠): Verifica si dos términos no son iguales.

Ejemplo: x≠yx ⋅ =y es verdadero si xx y yy no son iguales.

Menor que (<): Verifica si el primer término es menor que el segundo.

Ejemplo: x <y x< y es verdadero si xx es menor que yy.

Menor o igual que (≤): Verifica si el primer término es menor o igual que el segundo.

Ejemplo: x≤yx≤y es verdadero si xx es menor o igual que yy.

Mayor que (>): Verifica si el primer término es mayor que el segundo.

Ejemplo: x>yx>y es verdadero si xx es mayor que yy.

Mayor o igual que (≥): Verifica si el primer término es mayor o igual que el segundo.

Ejemplo: x≥yx≥y es verdadero si xx es mayor o igual que yy.

Combinación de Operadores

Los operadores lógicos, relacionales y aritméticos pueden combinarse en expresiones más complejas para evaluar condiciones y realizar cálculos.

Uso en Proposiciones

Los operadores relacionales son comúnmente utilizados en proposiciones que involucran variables y constantes.

Proposición: "Si la edad de Juan es mayor que 18, entonces Juan es adulto."

Notación: edad($Juan$)>18→adulto($Juan$)edad(Juan)>18→adulto(Juan)

Proposición: "Para todo número xx, si xx es menor que 10, entonces x+5x+5 es menor que 15."

Notación: ∀x (x<10→x+5<15) ∀ x (x<10→x+5<15)

Proposición: "Existe un número yy tal que yy es mayor o igual a 0 y menor que 5."

Notación: ∃ y (0≤y<5) ∃ y (0≤y<5)

Los algoritmos unidireccionales o secuenciales.

Definición de Algoritmos Secuenciales

Algoritmos Secuenciales: Son algoritmos en los que las instrucciones se ejecutan de manera secuencial, es decir, en un orden predefinido y sin ninguna bifurcación o repetición de pasos. Cada instrucción se ejecuta una sola vez, siguiendo el orden en que están escritas.

Características de los Algoritmos Secuenciales

Flujo Lineal: Las instrucciones se ejecutan una tras otra en un orden específico.

Simplicidad: Son fáciles de entender y seguir, ya que no incluyen estructuras complejas como bucles o condiciones.

Determinismo: El resultado es predecible y siempre será el mismo para las mismas entradas, ya que no hay variaciones en el flujo de ejecución.

Un algoritmo simple que toma dos números como entrada y devuelve su suma.

Pseudocódigo:

markdown — Copiar código

```
Inicio
    Leer número1
    Leer número2
    suma ← número1 + número2
    Escribir suma
Fin
```

Imagen N° 5. Ejemplo de Pseudocodigo
Fuente**: Chat GPT4 2024**

Un algoritmo que calcula el área de un rectángulo dados su ancho y alto.

Ejemplo de Uso de Operadores Aritméticos

pseudocode Copiar código

```
Proceso OperadoresAritmeticos
    Definir a, b Como Real
    Definir suma, resta, multiplicacion, division, modulo Como Real

    Escribir "Ingrese el valor de a: "
    Leer a
    Escribir "Ingrese el valor de b: "
    Leer b

    // Ejemplos de operadores aritméticos
    suma = a + b
    Escribir "a + b = ", suma

    resta = a - b
    Escribir "a - b = ", resta

    multiplicacion = a * b
    Escribir "a * b = ", multiplicacion

    division = a / b
    Escribir "a / b = ", division

    modulo = a % b
    Escribir "a % b = ", modulo
FinProceso
```

Imagen N° 6. Ejemplo de Pseudocodigo operadores matemáticos.
Fuente**: Chat GPT4 2024**

Algoritmos Condicionales

Algoritmos Condicionales: Son algoritmos que utilizan estructuras de decisión para seleccionar entre diferentes acciones basadas en la evaluación de condiciones lógicas. La estructura básica incluye la condición "si" (if), que puede ir acompañada de "si no" (else) y "si no, si" (elif o else if).

Componentes de un Algoritmo Condicional

Condición: Una expresión lógica que se evalúa como verdadera o falsa. Bloque de Código: El conjunto de instrucciones que se ejecutan si la condición

es verdadera. Alternativa (opcional): Un conjunto de instrucciones que se ejecutan si la condición es falsa.

Estructura de Algoritmo Condicional

Condicional Simple (if):

Se ejecuta un bloque de código si la condición es verdadera.

```
Inicio
    Si (condición) Entonces
        // Bloque de código
    Fin Si
Fin
```

Condicional con Alternativa (if-else):

Se ejecuta un bloque de código si la condición es verdadera y otro bloque si la condición es falsa.

```
Inicio
    Si (condición) Entonces
        // Bloque de código si la condición es verdadera
    Sino
        // Bloque de código si la condición es falsa
    Fin Si
Fin
```

Condicional Anidado (if-elif-else):

Permite múltiples condiciones y alternativas.

```
Inicio
    Si (condición1) Entonces
        // Bloque de código si condición1 es verdadera
    Sino Si (condición2) Entonces
        // Bloque de código si condición2 es verdadera
    Sino
        // Bloque de código si ninguna condición anterior es verdadera
    Fin Si
Fin
```

Ejemplo de Combinación de Operadores

pseudocode — Copiar código

```
Proceso CombinacionDeOperadores
    Definir edad Como Entero
    Definir salario Como Real
    Definir esAdulto, esRico, resultado Como Logico

    Escribir "Ingrese su edad: "
    Leer edad
    Escribir "Ingrese su salario mensual: "
    Leer salario

    // Evaluar si es adulto
    esAdulto = edad >= 18

    // Evaluar si es rico
    esRico = salario > 5000

    // Combinación de operadores lógicos y relacionales
    resultado = esAdulto Y esRico
    Escribir "¿Es adulto y rico? ", resultado

    resultado = esAdulto O esRico
    Escribir "¿Es adulto o rico? ", resultado

    resultado = NO(esAdulto)
    Escribir "¿No es adulto? ", resultado
FinProceso
```

Imagen N° 7 Combinacion de operadores
Fuente**: Chat GPT4 2024**

Unidad III

La estructura repetitiva Do-while

Es aquella en que el cuerpo del bucle se repite mientras que se cumple una determinada condición. En esta estructura, la condición del ciclo se evalúa al final, por lo que siempre se ejecutarán las instrucciones del ciclo por lo menos una vez.

Si la condición se evalúa verdadera, se ejecuta nuevamente el cuerpo del bucle; si la condición es falsa, entonces sale y se sigue con el flujo normal del algoritmo. Este proceso se repite una y otra vez hasta que la condición sea falsa.

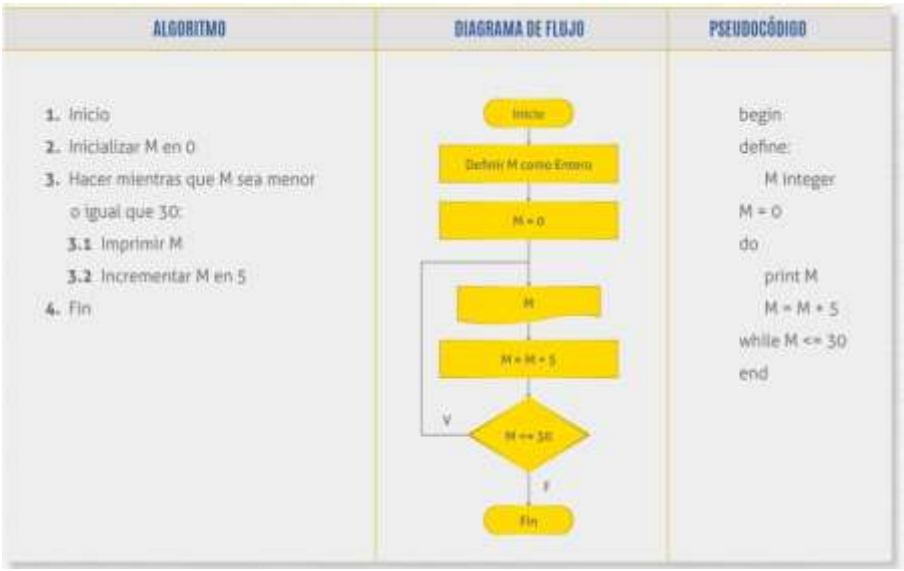


Imagen N°8. Ejemplo Algoritmo, diagrama de Flujo Pseudocodigo Ciclo.
Fuente: **Gustavo Jiménez 2024**

La estructura repetitiva Do-Until:

Es aquella en que el cuerpo del bucle se repite hasta que se cumple una determinada condición. En esta estructura, la condición del ciclo se evalúa al final, por lo que siempre se ejecutarán las instrucciones del ciclo por lo menos una vez. Si la condición se evalúa falsa, se ejecuta nuevamente el cuerpo del bucle; si la condición es verdadera, entonces sale y se sigue con el flujo normal del algoritmo. Este proceso se repite una y otra vez hasta que la condición sea verdadera.

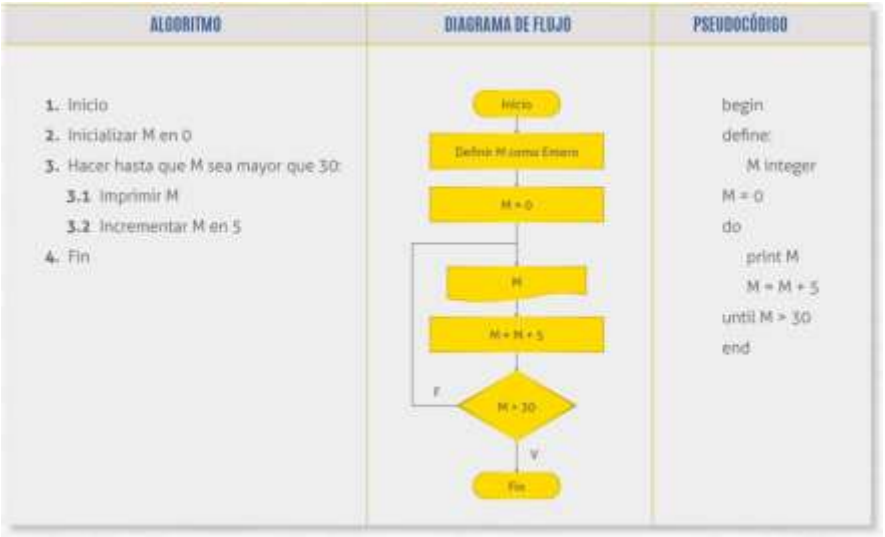


Imagen N°9. Ejemplo Algoritmo, diagrama de Flujo Pseudocodigo Ciclo.
Fuente: **Gustavo Jiménez 2024**

La estructura repetitiva while:

Es aquella en que el cuerpo del bucle se repite mientras se cumple una determinada condición. En esta estructura, la condición del ciclo se evalúa al inicio; si la condición se evalúa falsa, no se entra al ciclo y se sigue con el flujo normal del algoritmo; si la condición es verdadera, entonces se entra al ciclo y

se ejecuta el cuerpo del bucle (instrucciones dentro del mientras), después se evalúa de nuevo la expresión booleana. Este proceso se repite una y otra vez mientras la condición sea verdadera.

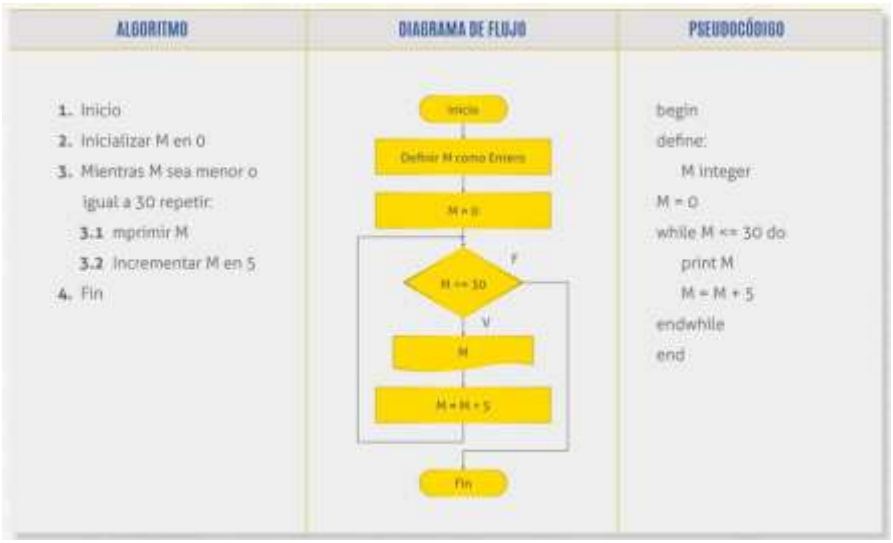


Imagen N°10. Ejemplo Algoritmo, diagrama de Flujo Pseudocodigo Ciclo.
Fuente: **Gustavo Jiménez 2024**

La estructura repetitiva For:

Este tipo de ciclos se utiliza cuando se conoce de antemano el número de veces que un conjunto de operaciones deberá ser ejecutado repetitivamente, generalmente utilizamos una variable de control como contador, que al llegar a un número predeterminado (condición de ejecución) termina la ejecución del ciclo, ordinariamente hacemos uso de las letras i, j, k, x, y para representar estas variables de control.

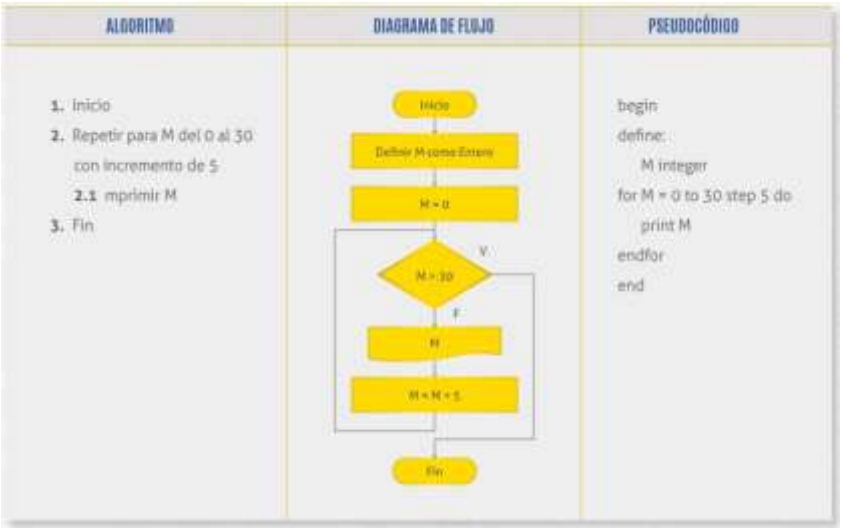


Imagen N°11. Ejemplo Algoritmo, diagrama de Flujo Pseudocodigo Ciclo.
Fuente: **Gustavo Jiménez 2024**
Introducción al Pseint.

Es un software que permite a los usuarios escribir, depurar y ejecutar pseudocódigo de manera interactiva. Proporciona un entorno amigable donde los principiantes pueden experimentar con algoritmos sin preocuparse por la sintaxis estricta de los lenguajes de programación tradicionales.

Beneficios de Usar PSeInt al Inicio de la Programación

Facilita el Aprendizaje de Conceptos Básicos:

Permite a los estudiantes concentrarse en la lógica y estructura de los algoritmos sin preocuparse por la sintaxis compleja de los lenguajes de programación.

Interfaz Intuitiva:

La interfaz gráfica de PSeInt es fácil de usar, lo que hace que sea accesible para aquellos que están comenzando en el mundo de la programación.

Depuración y Ejecución:

Los estudiantes pueden ejecutar y depurar su pseudocódigo, viendo cómo se comporta su algoritmo en tiempo real. Esto facilita la identificación y corrección de errores.

Biblioteca de Ejemplos:

PSeInt incluye una amplia biblioteca de ejemplos y ejercicios que los estudiantes pueden usar para practicar y aprender diferentes conceptos de programación.

Soporte Multiplataforma:

PSeInt está disponible para múltiples plataformas, incluyendo Windows, Linux y MacOS, lo que permite a los estudiantes usarlo en el sistema operativo de su elección.

Transición Sencilla a Lenguajes de Programación:

Al aprender a estructurar algoritmos en pseudocódigo, los estudiantes adquieren una base sólida que facilita la transición a lenguajes de programación como Python, Java, C++, etc.

Interfaz Gráfica UI

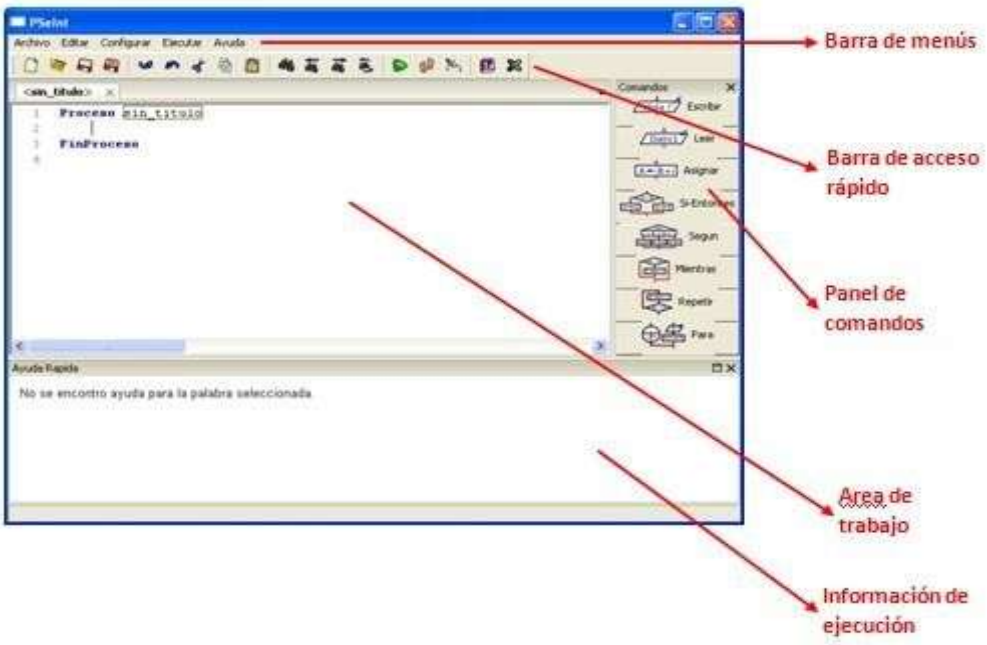


Imagen N°12. Ejemplo Algoritmo, diagrama de Flujo Pseudocodigo Ciclo.
Fuente: **Gustavo Jiménez 2024**

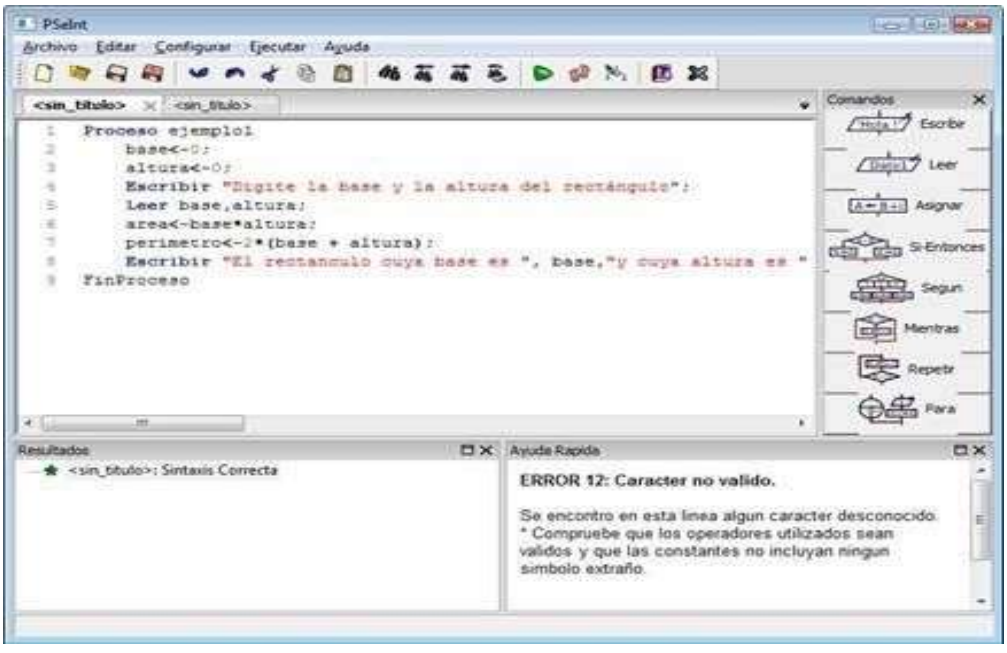


Imagen N°14. Ejemplo Algoritmo, diagrama de Flujo Pseudocodigo Ciclo.
Fuente: **Gustavo Jiménez 2024**

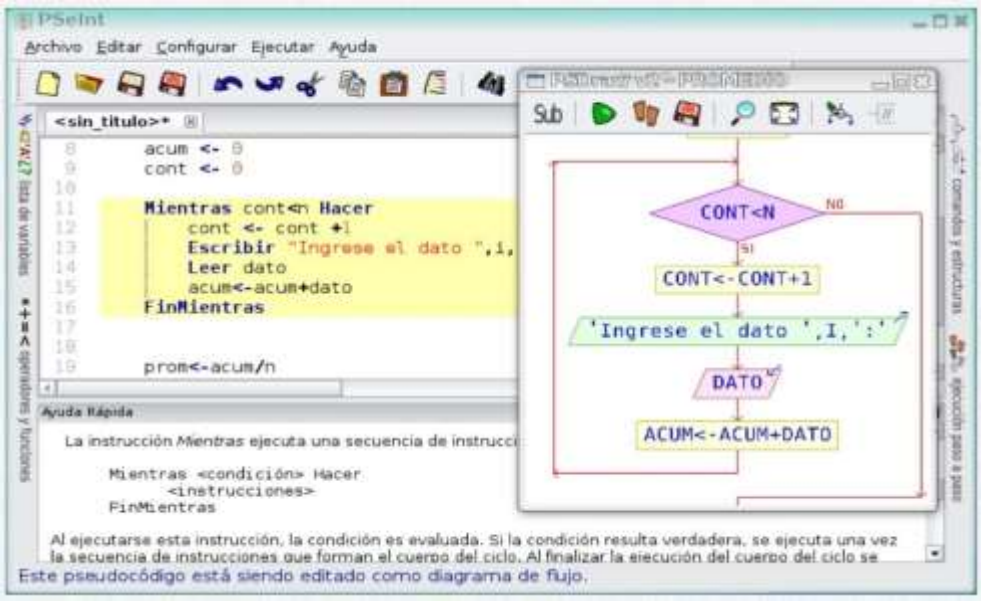


Imagen N°13. Ejemplo Algoritmo, diagrama de Flujo Pseudocodigo Ciclo.
Fuente: **Gustavo Jiménez 2024**

El uso didáctico del Pseint como Software Educativo

PSeInt le ayuda a escribir algoritmos utilizando un pseudo-lenguaje simple, intuitivo y en español.

Permite ejecutar el algoritmo para observar su funcionamiento y verificar los resultados.

Las reglas del lenguaje se pueden ajustar o flexibilizar según las necesidades de cada docente. El interprete incluye una lista de perfiles preconfigurados para las distintas instituciones que ya lo utilizan.

El editor ofrece diferentes tipos de ayudas mientras escribe (sugerencias, autocompletado, indentado, etc), y brinda la posibilidad de utilizar plantillas para los comandos básicos, junto con sus correspondientes descripciones que le ayudarán a completarlas

El intérprete identifica claramente los errores de su algoritmo y ofrece descripciones completas y sugerencias para que pueda corregirlos fácilmente.

Además del pseudocódigo, PSeInt permite trabajar con diagramas de flujo, convirtiendo automáticamente los algoritmos entre una y otra representación, siendo posible editarlos en ambos formatos.

El lenguaje incluye las estructuras de control más comunes, la posibilidad de definir funciones/subprocesos, y la capacidad de manipular arreglos de una o más dimensiones.

Puede ejecutar el algoritmo paso por paso para ver qué instrucciones se ejecutan y en qué orden, y observar cómo cambian los contenidos de las variables de su programa.

Dispone además de un modo de ejecución especial donde el intérprete explica en detalle la forma de procesar cada instrucción para que el alumno comprenda mejor tanto el funcionamiento del intérprete como la lógica del lenguaje.

El software es libre, gratuito y multiplataforma.

Incluye además una completa ayuda con ejemplos de diferentes niveles, y se actualiza frecuentemente para responder a las sugerencias y necesidades de los usuarios.

Lcdo. Gustavo Jiménez

Unidad IV

Estructuras condicionales, operaciones lógicas y relacionales.

En PSeInt, se pueden utilizar operadores lógicos para evaluar expresiones booleanas. Los operadores lógicos permiten combinar varias condiciones y tomar decisiones basadas en el resultado de esas combinaciones. Los operadores lógicos más comunes en PSeInt son:

1. **Y lógico (AND)**: Representado por Y.
2. **O lógico (OR)**: Representado por O.
3. **NO lógico (NOT)**: Representado por NO.

Ejemplo de Uso de Operadores Lógicos en PSeInt

A continuación, te presento un ejemplo simple de un programa en PSeInt que utiliza operadores lógicos para evaluar condiciones y tomar decisiones:

pseudocode — Copiar código

```
Proceso OperadoresLogicos
    Definir edad Como Entero
    Definir tieneLicencia Como Logico
    Definir puedeConducir Como Logico

    Escribir "Ingrese su edad: "
    Leer edad

    Escribir "¿Tiene licencia de conducir? (verdadero/falso): "
    Leer tieneLicencia

    // Evaluar si la persona puede conducir
    // Debe ser mayor de 18 años Y tener licencia de conducir
    puedeConducir = (edad >= 18) Y (tieneLicencia = verdadero)

    // Mostrar el resultado
    Si puedeConducir Entonces
        Escribir "Usted puede conducir."
    Sino
        Escribir "Usted no puede conducir."
    FinSi
FinProceso
```

Imagen N°15. Ejemplo Algoritmo Software Pseint Condicional.
Fuente: **Chat GPT 2024**

Uso del Operador `O` (OR):

pseudocode — Copiar código

```
Proceso UsoDeOperadorO
    Definir temperatura Como Real
    Definir esVerano Como Logico
    Definir haceCalor Como Logico

    Escribir "Ingrese la temperatura actual: "
    Leer temperatura

    Escribir "¿Es verano? (verdadero/falso): "
    Leer esVerano

    // Evaluar si hace calor
    // Hace calor si la temperatura es mayor a 30 O es verano
    haceCalor = (temperatura > 30) O (esVerano = verdadero)

    // Mostrar el resultado
    Si haceCalor Entonces
        Escribir "Hace calor."
    Sino
        Escribir "No hace calor."
    FinSi
FinProceso
```

Imagen N°16. Ejemplo Algoritmo Software Pseint Condicional.
Fuente: **Chat GPT 2024**

Practicas con operadores lógicos, matemáticos, relacionales con estructuras repetitivas

Las estructuras repetitivas en PSeInt permiten ejecutar un bloque de código varias veces según una condición específica. Estas estructuras son esenciales

para la programación, ya que permiten automatizar tareas repetitivas y manejar grandes conjuntos de datos de manera eficiente.

En PSeInt, las estructuras repetitivas más comunes son:

1. *Para:* Repite un bloque de código un número específico de veces.
2. *Mientras:* Repite un bloque de código mientras una condición sea verdadera.
3. *Repetir Hasta Que:* Repite un bloque de código hasta que una condición se vuelva verdadera.

Estructura Repetitiva "Para"

La estructura `Para` (o `For` en inglés) se utiliza cuando se conoce de antemano el número de iteraciones que se deben realizar.

Sintaxis: pseudocodigo

```
Para variable <- valorInicial Hasta valor Final Hacer
   // Bloque de código a repetir
FinPara
```

Ejemplo:

```
pseudocodigo Proceso
EjemploPara
   Definir i Como Entero
   Para i <- 1 Hasta 10 Hacer
      Escribir "Esta es la iteración: ", i
   FinPara
FinProceso
```

En este ejemplo, el bucle `Para` repite el bloque de código 10 veces, imprimiendo el número de iteración en cada ciclo.

Estructura Repetitiva "Mientras"

La estructura `Mientras` (o `While` en inglés) se utiliza cuando se quiere repetir un bloque de código mientras una condición sea verdadera. La condición se evalúa antes de cada iteración.

Ejemplo:

pseudocode — Copiar código

```
Proceso EjemploMientras
    Definir contador Como Entero
    contador <- 1

    Mientras contador <= 10 Hacer
        Escribir "Contador: ", contador
        contador <- contador + 1
    FinMientras
FinProceso
```

Imagen N°17. Ejemplo Algoritmo Software Pseint Ciclos.
Fuente: **Chat GPT 2024**

En este ejemplo, el bucle `Mientras` repite el bloque de código mientras el valor de `contador` sea menor o igual a 10.

Estructura Repetitiva "Repetir Hasta Que"

La estructura `Repetir Hasta Que` (o `Repeat Until` en inglés) se utiliza cuando se quiere repetir un bloque de código hasta que una condición se vuelva verdadera. La condición se evalúa después de cada iteración, por lo que el bloque de código se ejecuta al menos una vez.

Ejemplo:

pseudocode Copiar código

```
Proceso EjemploRepetirHastaQue
    Definir contador Como Entero
    contador <- 1

    Repetir
        Escribir "Contador: ", contador
        contador <- contador + 1
    Hasta Que contador > 10
FinProceso
```

Imagen N°18. Ejemplo Algoritmo Software Pseint Ciclos.
Fuente: **Chat GPT 2024**

En este ejemplo, el bucle `Repetir Hasta Que` repite el bloque de código hasta que el valor de `contador` sea mayor que 10.

pseudocode Copiar código

```
Proceso EstructurasRepetitivas
    Definir i, suma, contador Como Entero

    // Uso de Para
    Escribir "Estructura Para:"
    Para i <- 1 Hasta 5 Hacer
        Escribir "Iteración: ", i
    FinPara

    // Uso de Mientras
    suma <- 0
    contador <- 1
    Escribir "Estructura Mientras:"
    Mientras contador <= 5 Hacer
        suma <- suma + contador
        Escribir "Contador: ", contador, " Suma: ", suma
        contador <- contador + 1
    FinMientras

    // Uso de Repetir Hasta Que
    contador <- 1
    suma <- 0
    Escribir "Estructura Repetir Hasta Que:"
    Repetir
        suma <- suma + contador
        Escribir "Contador: ", contador, " Suma: ", suma
        contador <- contador + 1
    Hasta Que contador > 5
FinProceso
```

Imagen N°19. Ejemplo Algoritmo Software Pseint Ciclos.
Fuente: **Chat GPT 2024**

Lcdo. Gustavo Jiménez

Unidad V

En PSeInt, los vectores y matrices son arreglos (arrays) que permiten almacenar y manipular múltiples valores en una sola variable:

Vectores

Son arreglos de una fila por n columnas (vector de fila) o una columna por n filas (vector de columna). Almacenan elementos del mismo tipo en fila y permiten el acceso aleatorio rápido a cualquier elemento, así como agregar y eliminar elementos de la secuencia de forma dinámica. Para declarar e ingresar datos en un vector unidimensional en PSeInt, se puede usar la palabra reservada "Dimension" y ciclos "Para".

```
1  Proceso AplicacionDeFuncionAUnVector
2
3      Dimension vector[50]
4      Dimension vectorFuncion[50]
5      suma <- 0
6
7      Para x <- 1 Hasta 50 Hacer
8          vector[x] <- -100 + Azar(200)
9          suma <- suma + vector[x]
10     FinPara
11
12     Para x <- 1 Hasta 50 Hacer
13         vectorFuncion[x] <- ( (vector[x] ^ 2) + (3 * vector[x]) - 2 ) + (suma / 50)
14     FinPara
15
16     Para x <- 1 Hasta 50 Hacer
17         Escribir "F[",vector[x],"] = ",vectorFuncion[x]
18     FinPara
19
20 FinProceso
```

Imagen N°20. Ejemplo Algoritmo Software Pseint Vectores.
Fuente: **Gustavo Jiménez**

Matrices

Son arreglos de m filas por n columnas. Almacenan números enteros, reales o caracteres alfanuméricos, y cada elemento se ubica mediante coordenadas de fila y columna.

Operaciones*:* Los elementos de vectores y matrices pueden ser manipulados y combinados mediante bucles y operaciones aritméticas.

```
4       Definir matriz Como Entero;
5       Dimension matriz[4,4];
6       escribir " "
7       escribir "Realizar un algoritmo que permita crear una matriz 3X3, inserta
8       escribir " "
9       Para fila<-1 Hasta 3 Con Paso 1 Hacer
10          Para columna<-1 Hasta 3 Con Paso 1 Hacer
11              Escribir "Ingrese la celda" " ",fila, " " ,columna;
12              leer matriz[fila,columna];
13          fin Para
14      Fin Para
15
16      Para fila<-1 Hasta 3 Con Paso 1 Hacer
17          Para columna<-1 Hasta 3 Con Paso 1 Hacer
18              Escribir sin saltar matriz[fila,columna]," ";
19          fin Para
20          Escribir " ";
21      Fin Para
22
23  FinAlgoritmo
```

Imagen N°20. Ejemplo Algoritmo Software Pseint Matrices.
Fuente: **Gustavo Jiménez**

Suma de Matrices

pseudocode — Copiar código

```
Proceso SumaDeMatrices
    Definir matrizA[3, 3], matrizB[3, 3], matrizSuma[3, 3] Como Entero
    Definir i, j Como Entero

    // Leer los elementos de la primera matriz
    Para i <- 1 Hasta 3 Hacer
        Para j <- 1 Hasta 3 Hacer
            Escribir "Ingrese el elemento [", i, ", ", j, "] de la matriz A: "
            Leer matrizA[i, j]
        FinPara
    FinPara

    // Leer los elementos de la segunda matriz
    Para i <- 1 Hasta 3 Hacer
        Para j <- 1 Hasta 3 Hacer
            Escribir "Ingrese el elemento [", i, ", ", j, "] de la matriz B: "
            Leer matrizB[i, j]
        FinPara
    FinPara

    // Sumar los elementos de las matrices
    Para i <- 1 Hasta 3 Hacer
        Para j <- 1 Hasta 3 Hacer
            matrizSuma[i, j] = matrizA[i, j] + matrizB[i, j]
        FinPara
    FinPara

    // Imprimir la matriz resultante
    Escribir "Matriz resultado (Suma):"
    Para i <- 1 Hasta 3 Hacer
        Para j <- 1 Hasta 3 Hacer
            Escribir "Elemento [", i, ", ", j, "]: ", matrizSuma[i, j]
        FinPara
    FinPara
FinProceso
```

Imagen N°21. Ejemplo Algoritmo Software Pseint Suma de Matrices. Fuente: **Chat GPT 2024**

En Resumen:

Vectores: Arreglos unidimensionales que permiten almacenar secuencias de elementos del mismo tipo.

Matrices*:* Arreglos bidimensionales que permiten almacenar elementos en una tabla de filas y columnas.

Lcdo. Gustavo Jiménez

Examen Modelo.

Parte 1: Tablas de Verdad (30 puntos)

1. (10 puntos) Construye la tabla de verdad para la siguiente expresión lógica:

$$(p \land \neg q) \rightarrow (q \lor r)$$

Donde:

- p: verdadero
- q: falso
- r: verdadero

Completa la tabla de verdad mostrando los resultados de cada paso intermedio y la evaluación final.

2. (10 puntos) Dada la siguiente expresión, determina si es una tautología, una contradicción, o una contingencia:

$$(\neg p \lor q) \land (p \rightarrow q)$$

Completa la tabla de verdad y explica el tipo de proposición.

$$(p \lor q) \land (\neg p \lor r)$$

3. (10 puntos) Resuelve la tabla de verdad para la expresión lógica:

$$(p \lor q) \land (\neg p \lor r)$$

Asegúrate de mostrar paso por paso cada columna de la tabla.

Parte 2: Algoritmos en Pseudocódigo (40 puntos)

4. (15 puntos) Diseña un algoritmo en pseudocódigo que permita determinar si un número ingresado por el usuario es primo o no.

Requisitos:

- El algoritmo debe permitir al usuario ingresar el número.
- Debe iterar a través de los números menores que el número ingresado para verificar si es divisible solo por 1 y por sí mismo.

5. (10 puntos) Escribe un algoritmo en pseudocódigo que realice lo siguiente:

- Solicite al usuario ingresar tres números.
- Determine cuál de los tres es el mayor.
- Muestre el resultado al usuario.

6. (15 puntos) Diseña un algoritmo en pseudocódigo que sume todos los números pares entre 1 y 100.

Requisitos:

- Usa una estructura de repetición para iterar a través de los números.
- Asegúrate de que solo se sumen los números pares.
- Al finalizar, muestra el resultado de la suma.

Parte 3: Resolución de Problemas Lógicos (30 puntos)

7. (10 puntos) Si se sabe que:

- p es "El programa compila correctamente".
- q es "El algoritmo tiene un error".
- r es "El código está optimizado".

Escribe una expresión lógica para representar la siguiente oración:

"Si el programa no compila correctamente o el algoritmo tiene un error, entonces el código no está optimizado".

Luego, construye la tabla de verdad de dicha expresión.

8. (10 puntos) Resuelve el siguiente problema utilizando pseudocódigo:

- Se necesita calcular el promedio de los números positivos ingresados por el usuario. El usuario puede ingresar cualquier cantidad de números y cuando quiera dejar de ingresar más números, debe ingresar un número negativo. El algoritmo debe mostrar el promedio de los números ingresados al finalizar.

9. (10 puntos) A partir de las siguientes proposiciones, evalúa si son verdaderas o falsas usando una tabla de verdad:

- $(p \land q) \lor (\neg r)$
- $\neg(p \lor q) \rightarrow (r \land q)$

Donde:

- p: verdadero
- q: falso
- r: verdadero

Puntaje total: 100 puntos

Referencias Bibliográficas.

German E. Gallego, Jhon J. Ramírez M. 2019 Lógica y Resolución de Problemas, Universidad Francisco de Paula Santander San José de Cúcuta Santander Ecoe Ediciones: Universidad Francisco de Paula Santander, 2019.

OpenAI. (2024, 26 de julio). Fundamentos de Lógica por ChatGPT. ChatGPT [Modelo de lenguaje GPT-4]. Recuperado de https://chat.openai.com/

OpenAI. (2024, 26 de julio). Operadores Aritmeticos en Pseint por ChatGPT. ChatGPT [Modelo de lenguaje GPT-4]. Recuperado de https://chat.openai.com/

Patricio Celi V. 2023 Fundamentos de Programación en Pseint, Editorial Santiago del Castillo G.Quito.

Víctor Manolo, 2017 Ejercicios de sentencias condicionales en Pseint Recuperado el 12 de junio de 2024, https://victomanolo.wordpress.com/1415-2/

Yadira Corral de Franco 2014 Lógica Proposicional, Universidad de Carabobo, Campus Bárbula.

Yadira Corral de Franco, Luis H. Manzanares 2018, Lógica Matemática y Teoría de Conjuntos, Fondo Editorial OPSU, Caracas.