UNIVERSIDAD RURAL DE GUATEMALA

FACULTAD DE INGENIERÍA
Ingeniería en Sistemas con especialización en IA y Ciberseguridad

Bienvenida al curso • Práctica 1: análisis, pseudocódigo y expresiones

Del problema al algoritmo

Algoritmos • FE729 • Sesión 1 de 4

Ing. Mynor A. Cifuentes

Universidad Rural de Guatemala • Laboratorio presencial • 28 de septiembre de 2026

Agenda de hoy

Cuatro horas: primero nos conocemos, después escribimos el primer algoritmo

PRIMERA PARTE: BIENVENIDA

15 min	Presentación del docente y GitHub
15 min	La carrera: qué esperar
10 min	Generalidades del laboratorio

Hoy se trabaja la Práctica 1 del manual: unidades 1, 2 y 3 del programa (resolución de problemas, pseudocódigo y expresiones).

SEGUNDA PARTE: PRÁCTICA 1

30 min	**A.** Análisis del problema (EPS)
45 min	**B.** Primer algoritmo en PSeInt
15 min	Receso
35 min	**C.** Expresiones, `DIV` y `MOD`
35 min	**D.** Expresiones lógicas
25 min	Evaluación en Socrative
15 min	Dudas, respaldo y cierre

01 PARTE
ALGORITMOS

Bienvenida

¿Quién soy?



Ing. Mynor A. Cifuentes

Desarrollador de software

MynorCifuentes

mynorcifuentes.github.io

PERFIL

- Ingeniero en Ciencias y Sistemas, Universidad de San Carlos.
- Desarrollador de software: Python, JavaScript y Java.
- Intereses: compiladores, estructuras de datos y automatización.
- Este ciclo: laboratorios de **Algoritmos** y **Electrónica Digital**.

CÓMO TRABAJAREMOS

Pregunten sin miedo: **se escribe, se ejecuta, se corrige y se aprende**.

Mi GitHub

Donde guardo y comparto mis proyectos

66 repositorios públicos

6 lenguajes distintos

1 portafolio web



Escanee el código con su teléfono

- Ahí están los proyectos de mi carrera: tareas, prácticas y aplicaciones completas.
- Python, JavaScript, Java, C#, Visual Basic y HTML.
- Mi portafolio está publicado con **GitHub Pages**: `mynorcifuentes.github.io`

> Todo lo que ven ahí empezó igual que hoy: con un algoritmo escrito en papel.

Algunos proyectos y el curso de la carrera con el que se relacionan



JavaBridge JavaScript

Traductor que convierte código Java en código Python.

Compiladores

Sistema-Turnos-Medicos Python

Colas para gestionar turnos de atención, con interfaz Tkinter y gráficos en Graphviz.

Programación y estructuras de datos

Inventario-py Python

Aplicación web en Django que consume una API REST hecha en Flask.

Desarrollo web y bases de datos

TourneyJS JavaScript

Analizador léxico que interpreta archivos que describen un torneo deportivo.

Lenguajes y autómatas

Agricultura-Precision Python

Lee un XML, lo convierte en una matriz con listas enlazadas y la optimiza.

Programación II y III

GuateRiegos-2.0 Python

Optimización de un sistema de riego y de aplicación robótica de fertilizante.

Algoritmos e inteligencia artificial

¿Qué es GitHub y por qué crear su cuenta hoy?

GIT Y GITHUB

- **Git** guarda el historial de cambios de sus archivos: puede volver a cualquier versión anterior.
- **GitHub** es el sitio donde se publican esos proyectos para respaldarlos y trabajar en equipo.

GITHUB EDUCATION

Un estudiante verificado recibe, sin costo, herramientas profesionales y asistentes de programación con IA.

SU PORTAFOLIO

Al terminar la carrera, su GitHub mostrará lo que sabe hacer. Muchas empresas lo revisan antes que el currículum.

TAREA PARA ESTA SEMANA

1. Crear una cuenta en `github.com` con un usuario profesional (su nombre, no un apodo).
2. Crear el repositorio `Algoritmos-FE729`.
3. Subir ahí los archivos `.psc` de cada práctica.

02 PARTE
ALGORITMOS

La carrera de Ingeniería en Sistemas

¿Qué hace un ingeniero en sistemas?

Resuelve problemas reales con tecnología

ANALIZA

Entiende el problema de una empresa o comunidad y define qué necesita.

DISEÑA

Propone la solución: algoritmos, datos y la arquitectura del sistema.

CONSTRUYE

Programa, prueba y pone en funcionamiento el software.

AUTOMATIZA

Usa la inteligencia artificial para que los sistemas aprendan de los datos.

PROTEGE

Resguarda la información y los sistemas frente a ataques.

Su carrera: **Ingeniería en Sistemas con especialización en Inteligencia Artificial y Ciberseguridad**. Diez semestres que terminan con el grado de licenciatura y un trabajo de tesis.

Su ruta en la carrera

Lo que dice el pénsum de estudios

Ustedes están aquí

Semestres 1 y 2 Fundamentos	Semestres 3 a 5 Programación y datos	Semestres 6 y 7 Especialización	Semestres 8 y 9 Sistemas completos	Semestre 10 Cierre
Lógica de sistemas	Bases de datos I y II	Inteligencia artificial I y II	Desarrollo web	Tesis
Electrónica analógica	Programación I y II	Ciberseguridad I y II	Redes y comunicación I y II	Inteligencia artificial V
Algoritmos	Sistemas operativos	Programación III	Ingeniería de software	Ciberseguridad V
Electrónica digital	Lenguajes y autómatas	Arquitectura de computadoras	IA III y IV	Redes y comunicación III
Matemática I y II	Ingeniería en sistemas	Compiladores	Ciberseguridad III y IV	Tecnología de información
Física aplicada		Análisis de sistemas I y II	Administración de TI	

Durante toda la carrera: Inglés I a V, Matemática I a VI y Estadística, módulos de competitividad (varios impartidos en inglés) e Investigación y Proyectos I a IV, que conducen a la tesis.

¿Por qué Algoritmos es la base de todo?



Un modelo de **inteligencia artificial** es un algoritmo que aprende a partir de los datos. Un **cifrado** de ciberseguridad también es un algoritmo.

Esta tarde, en **Electrónica Digital**, verán cómo la computadora ejecuta esos algoritmos con ceros, unos y compuertas lógicas.

¿Qué esperar de la carrera?

Sin rodeos

MUCHA PRÁCTICA

A programar se aprende programando. Leer no basta: hay que escribir, equivocarse y corregir.

INGLÉS

Cinco cursos de inglés y módulos impartidos en inglés. La documentación técnica está, casi toda, en inglés.

APRENDER A APRENDER

La tecnología cambia cada año. Lo más valioso es saber aprender por cuenta propia.

MATEMÁTICA

Seis cursos de matemática y uno de estadística. La IA se apoya en álgebra, probabilidad y cálculo.

ERRORES NORMALES

Los errores son parte del trabajo. Depurar, es decir, encontrar y corregir fallas, es una habilidad que se entrena.

INVESTIGACIÓN

Investigación y Proyectos I a IV: monografía, tesina y tesis. Se aprende a documentar y a presentar resultados.

¿Dónde pueden trabajar?

Desarrollo de software
aplicaciones web, móviles y de escritorio

Inteligencia artificial y datos
modelos, análisis y automatización

Ciberseguridad
auditoría, defensa y respuesta a incidentes

Bases de datos
diseño y administración

Redes e infraestructura
servidores y servicios en la nube

Análisis y gestión de TI
proyectos tecnológicos en empresas

La especialización en IA y ciberseguridad los prepara para dos de las áreas con más demanda en la industria tecnológica.

Hábitos que marcan la diferencia

1. **Practique un poco cada día.** Treinta minutos diarios rinden más que cinco horas la noche antes del examen.
2. **Piense antes de teclear.** Primero el análisis y el algoritmo en papel; después la computadora.
3. **Pregunte pronto.** Una duda de hoy es un problema grande dentro de tres semanas.
4. **Colabore, no copie.** Explicarle a un compañero es la mejor forma de aprender; copiar se califica con cero.
5. **Use la IA como tutor.** Pídale que le explique, no que le resuelva. En Socrative estará usted solo.
6. **Guarde y publique su trabajo.** Carpetas ordenadas y todo en GitHub.

RECOMENDACIÓN

Ser ingeniero no es saberlo todo: es saber resolver problemas de forma ordenada. Justo eso es un algoritmo.

Ahora ustedes

Una presentación rápida de cada uno

EN 30 SEGUNDOS

1. Su nombre y la sede de donde viene.
2. ¿Por qué eligió Ingeniería en Sistemas?
3. ¿Ya ha programado alguna vez? ¿En qué?
4. Algo que le gustaría construir con lo que aprenda en la carrera.

No hay respuestas incorrectas.

03 PARTE
ALGORITMOS

Generalidades del laboratorio

El laboratorio de Algoritmos

Cuatro sesiones presenciales de cuatro horas, de lunes a jueves

Ses.	Tema central	Contenidos
1	**Del problema al algoritmo** hoy	Etapas de la resolución de problemas, análisis EPS, constantes, variables, tipos de datos, expresiones aritméticas y lógicas
2	Estructuras de control y arreglos	`Si`, `Segun`, ciclos `Mientras`, `Repetir` y `Para`; un arreglo de cinco notas
3	Diseño descendente y modularización	Procedimientos, funciones, parámetros por valor y por referencia
4	Proyecto guiado de las unidades 1 a 6	Menú, arreglo de notas, carga, listado, promedio y aprobados

NOTA

Cubre las unidades 1 a 6 del programa (hasta el segundo parcial). La clase magistral continúa en línea.

¿Cómo se evalúa cada práctica?



60 PUNTOS: TRABAJO EN EL LABORATORIO

Seguir el procedimiento, completar las tablas y lograr que los algoritmos se ejecuten.

40 PUNTOS: SOCRATIVE

Evaluación individual al final de la sesión. Combina teoría con lo que se hizo en la práctica.

NOTA

No se entrega informe. La nota se gana aquí, durante la jornada.

Herramientas y organización

MATERIALES Y EQUIPO

- Computadora con **PSeInt** instalado.
- Cuaderno cuadriculado, lápiz y borrador.
- Memoria USB o una carpeta en la nube.
- Navegador con internet para **Socrative**.

NOTA

PSeInt ejecuta pseudocódigo en español, dibuja el diagrama de flujo y explica los errores en español.

ORGANICE SUS ARCHIVOS

```
Algoritmos_FE729/
  Practica_01/
    P01_nota.psc
    P01_expresiones.psc
    P01_tiempo.psc
    P01_logicas.psc
  Practica_02/
```

RECOMENDACIÓN

El orden de los archivos es parte de la nota de *trabajo en el laboratorio*.

Normas del laboratorio

NORMAS

1. Puntualidad: el tiempo perdido no se repone.
2. Leer la práctica *antes* de la sesión.
3. No instalar programas ni cambiar la configuración de los equipos.
4. Guardar con frecuencia y respaldar antes de salir.
5. Área de trabajo limpia; al final, equipo apagado.
6. Se trabaja solo o en parejas, según se indique.
7. Socrative es **individual**.
8. Copiar sin citar la fuente se califica con **cero**.

04 PARTE
ALGORITMOS

Práctica 1: Del problema al algoritmo

Objetivos de la práctica

OBJETIVOS

- Completar una tabla de entradas, proceso y salidas para un problema corto.
- Distinguir una constante de una variable.
- Copiar, guardar y ejecutar un algoritmo en PSeInt.
- Calcular a mano y en PSeInt expresiones con `+`, `*`, `DIV` y `MOD`.
- Evaluar expresiones con `Y`, `O` y `NO`.

UNIDADES DEL PROGRAMA

1. Etapas de la resolución de problemas
2. Elementos del pseudocódigo
3. Expresiones

> Ritmo pausado: **se escribe, se ejecuta y se anota** el resultado.

¿Qué es un algoritmo?

Un **algoritmo** es una secuencia **finita**, **ordenada** y **no ambigua** de instrucciones que resuelve un problema determinado.

FINITO

Termina después de un número limitado de pasos.

ORDENADO

Cada paso tiene su lugar: cambiar el orden cambia el resultado.

NO AMBIGUO

Cada instrucción significa una sola cosa. No hay que adivinar.

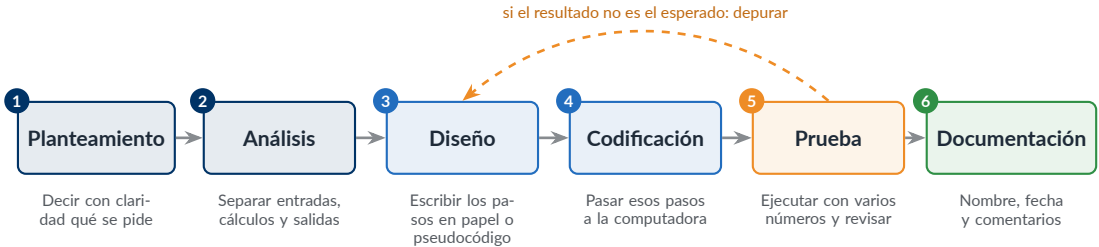
ESTO NO ES UN ALGORITMO

"Póngale un poco de azúcar al café y muévalo un rato."
¿Cuánto es *un poco*? ¿Cuánto dura *un rato*?

ESTO SÍ

1. Agregue 2 cucharaditas de azúcar. 2. Mueva 10 veces.
3. Pruebe: si está simple, repita el paso 1 con 1 cucharadita.

Las seis etapas de la resolución de problemas

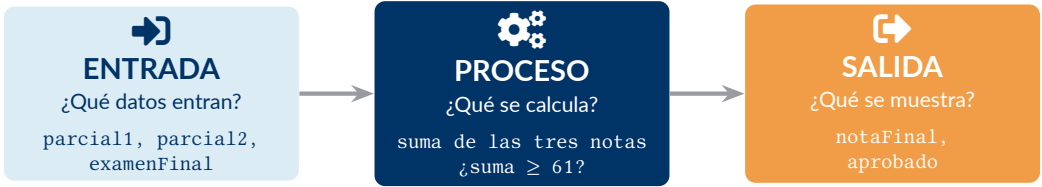


Depurar es buscar y corregir los errores que aparecen en la etapa de prueba.

Documentación interna: comentarios dentro del algoritmo.
Documentación externa: manuales y explicaciones aparte.

Análisis Entrada – Proceso – Salida (EPS)

Tres preguntas antes de escribir una sola instrucción



RECOMENDACIÓN

Si no puede llenar estas tres cajas, todavía no entiende el problema. Vuelva a leerlo antes de abrir PSeInt.

El problema de hoy

Se usará el mismo problema durante toda la sesión

Problema. Un estudiante tiene dos parciales. Cada parcial vale como máximo **30 puntos**. El examen final vale como máximo **40 puntos**. El algoritmo debe sumar las tres notas y decir si aprobó. Se aprueba con **61 puntos o más**.

PRIMERA LECTURA

Marque **qué se pide**: la nota final y si aprobó o no.

SEGUNDA LECTURA

Marque **con qué números se calcula**: dos parciales y un examen, cada uno con su rango.

NOTA

La tabla EPS se llena **en el cuaderno**, todavía sin abrir PSeInt. El teclado viene después.

Parte A. Tabla de análisis EPS

30 min

Dato	Clasificación	Tipo	Rango
Primer parcial	Entrada		0 a 30
Segundo parcial	Entrada		0 a 30
Examen final	Entrada		0 a 40
Suma de las tres notas	Proceso	—	—
Nota final	Salida		
Aprobó (sí o no)	Salida		

NOTA

Complete las columnas vacías en su cuaderno. Si duda del tipo, pregunte antes de seguir.

Parte A. Tabla de análisis EPS

30 min

Dato	Clasificación	Tipo	Rango
Primer parcial	Entrada	`Real`	0 a 30
Segundo parcial	Entrada	`Real`	0 a 30
Examen final	Entrada	`Real`	0 a 40
Suma de las tres notas	Proceso	—	—
Nota final	Salida	`Real`	0 a 100
Aprobó (sí o no)	Salida	`Logico`	`Verdadero` o `Falso`

RECOMENDACIÓN

Las notas pueden tener decimales (por ejemplo, 15.5), por eso se usa `Real`. Si el curso solo usara enteros, `Entero` también sería válido.

Tipos de datos

Tipo	Almacena	Ejemplo
`Entero`	Número sin decimales	`0`, `61`, `100`
`Real`	Número con decimales	`15.5`, `3.14`
`Cadena`	Texto	`"Ana"`
`Logico`	Verdadero o falso	`Verdadero`

¿QUÉ TIPO USARÍA?

- La edad de una persona
- El precio de un libro
- El nombre de su sede
- ¿Es mayor de edad?
- Cantidad de estudiantes

El tipo define **qué valores** puede guardar un dato y **qué operaciones** se pueden hacer con él: no se puede multiplicar un nombre.

Tipos de datos

Tipo	Almacena	Ejemplo
`Entero`	Número sin decimales	`0`, `61`, `100`
`Real`	Número con decimales	`15.5`, `3.14`
`Cadena`	Texto	`"Ana"`
`Logico`	Verdadero o falso	`Verdadero`

¿QUÉ TIPO USARÍA?

- La edad de una persona — `Entero`
- El precio de un libro — `Real`
- El nombre de su sede — `Cadena`
- ¿Es mayor de edad? — `Logico`
- Cantidad de estudiantes — `Entero`

El tipo define **qué valores** puede guardar un dato y **qué operaciones** se pueden hacer con él: no se puede multiplicar un nombre.

Constantes y variables

VARIABLE

Un espacio con nombre en la memoria cuyo valor **puede cambiar** mientras el algoritmo se ejecuta.

```
notaFinal <- parcial1 + parcial2 + examenFinal
```

BUENOS NOMBRES

- Empiezan con una letra, sin espacios ni tildes.
- Dicen qué guardan: `notaFinal` y no `x`.
- Constantes en mayúsculas: `NOTA_MINIMA`.

CONSTANTE

Un valor con nombre que **no cambia** durante la ejecución.

```
NOTA_MINIMA <- 61
```

¿Por qué usar una constante? Si el reglamento cambia la nota mínima a 65, se modifica **una sola línea** y el resto del algoritmo sigue funcionando.

Parte B. El primer algoritmo en PSeInt

Guárdelo como `P01_nota.psc`

45 min

```
1  Algoritmo CalculoNota
2    // Universidad Rural de Guatemala - Algoritmos FE729
3    // Proposito: sumar dos parciales y el examen final
4    // Declaracion de variables
5    Definir NOTA_MINIMA Como Entero
6    Definir parcial1, parcial2, examenFinal, notaFinal Como Real
7    Definir aprobado Como Logico
8    // Inicializacion
9    NOTA_MINIMA <- 61
10   Escribir "Primer parcial (0 a 30):"
11   Leer parcial1
12   Escribir "Segundo parcial (0 a 30):"
13   Leer parcial2
14   Escribir "Examen final (0 a 40):"
15   Leer examenFinal
16   // Calculos
17   notaFinal <- parcial1 + parcial2 + examenFinal
18   aprobado <- notaFinal >= NOTA_MINIMA
19   // Resultados
20   Escribir "Nota final: ", notaFinal
21   Escribir "Aprobado: ", aprobado
22 FinAlgoritmo
```

Líneas	Parte
1, 22	**Inicio y cierre:** `Algoritmo` … `FinAlgoritmo`
2–3	**Documentación interna**
5–7	**Variables:** `Definir` … `Como`
9	**Constante:** se asigna una sola vez
10–15	**Entrada**: `Leer`
17–18	**Proceso**: `<-` asigna
20–21	**Salida**: `Escribir`

> La flecha `<-` se lee *"recibe"*: *notaFinal recibe la suma*. Se teclea con `<` y `-`.

Otras notaciones y PSeInt

En libros y en la clase en línea puede aparecer otra notación: así se traduce

Notación general	En PSeInt
`Const NOTA_MINIMA <- 61`	`Definir NOTA_MINIMA Como Entero` y después `NOTA_MINIMA <- 61`
`Var Real : x`	`Definir x Como Real`
`Logico : aprobado`	`Definir aprobado Como Logico`
`Inicio` … `Fin`	No se escriben: todo va entre `Algoritmo` y `FinAlgoritmo`
≥ ≤ ≠	`>=` `<=` `<>`
`a DIV b`	`trunc(a / b)`
`a MOD b`	`a MOD b` (igual)

ADVERTENCIA

Si copia un algoritmo de un libro o de internet, **tradúzcalo** antes de ejecutarlo. El manual del laboratorio ya usa la sintaxis de PSeInt (anexo A).

NOTA

Los símbolos ≥, ≤ y ≠ no están en el teclado: se escriben con dos caracteres. Si su perfil de PSeInt es estricto, termine cada instrucción con `;`.

Probar el primer algoritmo

Etapa 5: ejecutar con varios datos y comparar con el cálculo a mano

Datos	A mano	Nota final	Aprobado	¿Por qué?
20, 25, 30	$20 + 25 + 30 = 75$			
20, 20, 20	$20 + 20 + 20 = 60$			
20, 21, 20	$20 + 21 + 20 = 61$			

Complete en su cuaderno: *NOTA_MINIMA es una constante porque* _____.

RECOMENDACIÓN

Pruebe siempre el **caso límite**. Los errores se esconden en el 61, no en el 75.

Probar el primer algoritmo

Etapa 5: ejecutar con varios datos y comparar con el cálculo a mano

Datos	A mano	Nota final	Aprobado	¿Por qué?
20, 25, 30	$20 + 25 + 30 = 75$	75	VERDADERO	75 es mayor o igual que 61
20, 20, 20	$20 + 20 + 20 = 60$			
20, 21, 20	$20 + 21 + 20 = 61$			

Complete en su cuaderno: *NOTA_MINIMA es una constante porque* _____.

RECOMENDACIÓN

Pruebe siempre el **caso límite**. Los errores se esconden en el 61, no en el 75.

Probar el primer algoritmo

Etapa 5: ejecutar con varios datos y comparar con el cálculo a mano

Datos	A mano	Nota final	Aprobado	¿Por qué?
20, 25, 30	$20 + 25 + 30 = 75$	75	VERDADERO	75 es mayor o igual que 61
20, 20, 20	$20 + 20 + 20 = 60$	60	FALSO	60 es menor que 61
20, 21, 20	$20 + 21 + 20 = 61$			

Complete en su cuaderno: *NOTA_MINIMA es una constante porque* _____.

RECOMENDACIÓN

Pruebe siempre el **caso límite**. Los errores se esconden en el 61, no en el 75.

Probar el primer algoritmo

Etapa 5: ejecutar con varios datos y comparar con el cálculo a mano

Datos	A mano	Nota final	Aprobado	¿Por qué?
$20,\ 25,\ 30$	$20 + 25 + 30 = 75$	75	VERDADERO	75 es mayor o igual que 61
$20,\ 20,\ 20$	$20 + 20 + 20 = 60$	60	FALSO	60 es menor que 61
$20,\ 21,\ 20$	$20 + 21 + 20 = 61$	61	VERDADERO	Justo en el límite: $61 \geq 61$

Complete en su cuaderno: *NOTA_MINIMA es una constante porque* _____.

RECOMENDACIÓN

Pruebe siempre el **caso límite**. Los errores se esconden en el 61, no en el 75.

Operadores aritméticos y orden de evaluación

Orden	Operadores	Ejemplo
1	`( )`	`(7 + 3) * 2` vale `20`
2	`* / DIV MOD`	`7 + 3 * 2` vale `13`
3	`+ -`	`10 - 4` vale `6`
4	`< <= > >= = <>`	`61 >= 61` es `Verdadero`
5	`Y`	las dos deben cumplirse
6	`O`	basta con que se cumpla una

PASO A PASO

`7 + 3 * 2`

↓ primero `*`

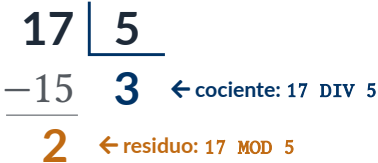
`7 + 6`

↓ después `+`

`13`

RECOMENDACIÓN

Si duda, use paréntesis. No cuestan nada y evitan errores.



`DIV` da el **cociente entero**: `17 DIV 5` vale `3`.
`MOD` da el **residuo**: `17 MOD 5` vale `2`.

¿PARA QUÉ SIRVEN?

- Minutos a horas: `130 DIV 60` = 2 y `130 MOD 60` = 10.
- Saber si un número es par: `n MOD 2 = 0`.
- Obtener el último dígito: `n MOD 10`.

Evalúe primero a mano y después en PSeInt

Archivo `P01_expresiones.psc`

No.	Expresión	Resultado
1	`7 + 3 * 2`	
2	`(7 + 3) * 2`	
3	`20 / 4`	
4	`20 DIV 3`	
5	`20 MOD 3`	
6	`17 MOD 5 + 17 DIV 5 *`	

```
Algoritmo Expresiones
  Escribir "1) ", 7 + 3 * 2
  Escribir "2) ", (7 + 3) * 2
  Escribir "3) ", 20 / 4
  Escribir "4) ", trunc(20 / 3)
  Escribir "5) ", 20 MOD 3
  Escribir "6) ", 17 MOD 5 + trunc(17 / 5) * 5
FinAlgoritmo
```

NOTA

En PSeInt, `a DIV b` se escribe `trunc(a / b)`: `trunc` elimina los decimales.

Evalúe primero a mano y después en PSeInt

Archivo `P01_expresiones.psc`

No.	Expresión	Resultado
1	`7 + 3 * 2`	13
2	`(7 + 3) * 2`	20
3	`20 / 4`	5
4	`20 DIV 3`	6
5	`20 MOD 3`	2
6	`17 MOD 5 + 17 DIV 5 *`	17

```
1 Algoritmo Expresiones
2   Escribir "1) ", 7 + 3 * 2
3   Escribir "2) ", (7 + 3) * 2
4   Escribir "3) ", 20 / 4
5   Escribir "4) ", trunc(20 / 3)
6   Escribir "5) ", 20 MOD 3
7   Escribir "6) ", 17 MOD 5 + trunc(17 / 5) * 5
8 FinAlgoritmo
```

NOTA

En PSeInt, `a DIV b` se escribe `trunc(a / b)`: `trunc` elimina los decimales.

Evalúe primero a mano y después en PSeInt

Archivo `P01_expresiones.psc`

No.	Expresión	Resultado
1	`7 + 3 * 2`	13
2	`(7 + 3) * 2`	20
3	`20 / 4`	5
4	`20 DIV 3`	6
5	`20 MOD 3`	2
6	`17 MOD 5 + 17 DIV 5 *`	17

Expresión 6: `2 + 3 * 5` → `2 + 15` → `17`

```
1 Algoritmo Expresiones
2   Escribir "1) ", 7 + 3 * 2
3   Escribir "2) ", (7 + 3) * 2
4   Escribir "3) ", 20 / 4
5   Escribir "4) ", trunc(20 / 3)
6   Escribir "5) ", 20 MOD 3
7   Escribir "6) ", 17 MOD 5 + trunc(17 / 5) * 5
8 FinAlgoritmo
```

NOTA

En PSeInt, `a DIV b` se escribe `trunc(a / b)`: `trunc` elimina los decimales.

Minutos a horas con DIV y MOD

Archivo P01_tiempo.psc

```
1  Algoritmo DescomponerTiempo
2    // Convierte minutos en horas y minutos
3    Definir totalMinutos, horas, minutos Como Entero
4    Escribir "Cantidad de minutos:"
5    Leer totalMinutos
6    // Calculos
7    horas <- trunc(totalMinutos / 60)
8    minutos <- totalMinutos MOD 60
9    // Resultados
10   Escribir horas, " hora(s) y ", minutos, " minuto(s)"
11 FinAlgoritmo
```

LAS DOS LÍNEAS CLAVE

`trunc(totalMinutos / 60)`: cuántas horas completas caben (el `DIV` de otros libros).

`totalMinutos MOD 60`: los minutos que sobran.

NOTA

Con `130` debe mostrar `2 hora(s) y 10 minuto(s)`.

Pruebas de `DescomponerTiempo`

Pruebe con 130, con 60 y con 59 minutos

Entrada	Horas	Minutos
130		
60		
59		

Comprobación: $130 = 2 \times 60 + 10$.
El cociente es `DIV` y el residuo es `MOD`.

Pruebas de `DescomponerTiempo`

Pruebe con 130, con 60 y con 59 minutos

Entrada	Horas	Minutos
130	2	10
60		
59		

Comprobación: $130 = 2 \times 60 + 10$.
El cociente es `DIV` y el residuo es `MOD`.

Pruebas de `DescomponerTiempo`

Pruebe con 130, con 60 y con 59 minutos

Entrada	Horas	Minutos
130	2	10
60	1	0
59		

Comprobación: $130 = 2 \times 60 + 10$.
El cociente es `DIV` y el residuo es `MOD`.

Pruebas de `DescomponerTiempo`

Pruebe con 130, con 60 y con 59 minutos

Entrada	Horas	Minutos
130	2	10
60	1	0
59	0	59

Comprobación: $130 = 2 \times 60 + 10$.
El cociente es `DIV` y el residuo es `MOD`.

RECOMENDACIÓN

Con `59`, `59 DIV 60` vale `0`: todavía no se completa una hora. Los casos `60` y `59` son los **límites** de este problema.

Operadores lógicos

Trabajan con valores `Verdadero` y `Falso`

`Y`: las dos deben cumplirse

p	q	p Y q
V	V	V
V	F	F
F	V	F
F	F	F

`O`: basta con una

p	q	p O q
V	V	V
V	F	V
F	V	V
F	F	F

`NO`: lo contrario

p	NO p
V	F
F	V

Ejemplo cotidiano: *"Aprueba si su nota es mayor o igual a 61* ***Y*** *tiene 80 % de asistencia."*

ADVERTENCIA

`Y` se evalúa antes que `O`. Si una expresión mezcla ambos, use paréntesis.

Parte D. Expresiones lógicas

Con a = 5, b = 10 y c = 5

35 min

No.	Expresión	Resultado
1	a = c	
2	a <> b	
3	a < b Y b > c	
4	a > b O b > c	
5	NO (a = c)	
6	a + 5 = b Y a = c	

```
Algoritmo Logicas
  Definir a, b, c Como Entero
  a <- 5
  b <- 10
  c <- 5
  Escribir "1) ", a = c
  Escribir "2) ", a <> b
  Escribir "3) ", a < b Y b > c
  Escribir "4) ", a > b O b > c
  Escribir "5) ", NO (a = c)
  Escribir "6) ", a + 5 = b Y a = c
FinAlgoritmo
```

Parte D. Expresiones lógicas

Con a = 5, b = 10 y c = 5

35 min

No.	Expresión	Resultado
1	a = c	Verdadero
2	a <> b	Verdadero
3	a < b Y b > c	Verdadero
4	a > b O b > c	Verdadero
5	NO (a = c)	Falso
6	a + 5 = b Y a = c	Verdadero

```
Algoritmo Logicas
  Definir a, b, c Como Entero
  a <- 5
  b <- 10
  c <- 5
  Escribir "1) ", a = c
  Escribir "2) ", a <> b
  Escribir "3) ", a < b Y b > c
  Escribir "4) ", a > b O b > c
  Escribir "5) ", NO (a = c)
  Escribir "6) ", a + 5 = b Y a = c
FinAlgoritmo
```

Parte D. Expresiones lógicas

Con a = 5, b = 10 y c = 5

35 min

No.	Expresión	Resultado
1	a = c	Verdadero
2	a <> b	Verdadero
3	a < b Y b > c	Verdadero
4	a > b O b > c	Verdadero
5	NO (a = c)	Falso
6	a + 5 = b Y a = c	Verdadero

Expresión 4: `5 > 10` es `Falso`, pero `10 > 5` es `Verdadero`. Con `O` basta una: el resultado es `Verdadero`.

```
1  Algoritmo Logicas
2    Definir a, b, c Como Entero
3    a <- 5
4    b <- 10
5    c <- 5
6    Escribir "1) ", a = c
7    Escribir "2) ", a <> b
8    Escribir "3) ", a < b Y b > c
9    Escribir "4) ", a > b O b > c
10   Escribir "5) ", NO (a = c)
11   Escribir "6) ", a + 5 = b Y a = c
12 FinAlgoritmo
```

Validar los datos de entrada

Regrese a `P01_nota.psc`

```
1  // Junto a los demas Definir:
2  Definir datosOk Como Logico
3  // Despues de Leer examenFinal:
4  datosOk <- (parcial1 >= 0) Y (parcial1 <= 30)
5  datosOk <- datosOk Y (parcial2 >= 0) Y (parcial2 <= 30)
6  datosOk <- datosOk Y (examenFinal >= 0) Y (examenFinal <= 40)
7  Escribir "Datos validos: ", datosOk
```

NOTA

Cada línea agrega una revisión con `Y`: `datosOk` solo sigue en `Verdadero` si todas se cumplen.

Datos	datosOk
20, 25, 30	
35, 25, 30	
20, 25, 45	

Cada paréntesis revisa un rango. Basta con que **uno** falle para que todo sea `Falso`, porque se unen con `Y`.

Validar los datos de entrada

Regrese a `P01_nota.psc`

```
1  // Junto a los demas Definir:
2  Definir datosOk Como Logico
3  // Despues de Leer examenFinal:
4  datosOk <- (parcial1 >= 0) Y (parcial1 <= 30)
5  datosOk <- datosOk Y (parcial2 >= 0) Y (parcial2 <= 30)
6  datosOk <- datosOk Y (examenFinal >= 0) Y (examenFinal <= 40)
7  Escribir "Datos validos: ", datosOk
```

NOTA

Cada línea agrega una revisión con `Y`: `datosOk` solo sigue en `Verdadero` si todas se cumplen.

Datos	datosOk
20, 25, 30	VERDADERO
35, 25, 30	FALSO
20, 25, 45	FALSO

Cada paréntesis revisa un rango. Basta con que **uno** falle para que todo sea `Falso`, porque se unen con `Y`.

Si PSeInt marca un error

Etapa de depuración

ERRORES FRECUENTES

- Falta `FinAlgoritmo` al final.
- Comillas sin cerrar: `"Nota final:`
- Falta la coma entre textos y variables en `Escribir`.
- Variable no definida o escrita distinto: `notafinal` y `notaFinal`.
- Usar `Const`, `Var`, `Inicio`, `DIV` o el símbolo ≥ en lugar de `>=`.

RECOMENDACIÓN

Lea el mensaje: está en español e **indica el número de línea**. Empiece a revisar por esa línea.

DIAGRAMA DE FLUJO

PSeInt dibuja el diagrama de flujo de su algoritmo con un botón. Úselo para comprobar que los pasos siguen el orden correcto.

Evaluación en Socrative

25 min Individual • 40 puntos

EVALUACIÓN EN LÍNEA (SOCRATIVE)

1. Abra `socrative.com` en el navegador o en la aplicación.
2. Elija **Iniciar sesión** y luego **Estudiante**.
3. Escriba el nombre de la sala.
4. Ingrese su **nombre completo**, tal como aparece en la lista.

NOMBRE DE LA SALA



NOTA

Las preguntas combinan la teoría de hoy con lo que usted hizo en PSeInt: los resultados de sus tablas cuentan.

Resumen de hoy

- Un **algoritmo** es finito, ordenado y no ambiguo.
- Antes de programar: **análisis EPS**.
- Seis etapas: planteamiento, análisis, diseño, codificación, prueba y documentación.
- Tipos básicos: `Entero`, `Real`, `Cadena` y `Logico`.
- Una **constante** no cambia; una **variable** sí.
- Primero paréntesis, luego `*` `/` `DIV` `MOD`, luego `+` `-`.
- `DIV`: cociente entero. `MOD`: residuo.
- `Y` exige ambas condiciones; `O`, al menos una; `NO` invierte.

RECOMENDACIÓN

Antes de salir: copie la carpeta `Practica_01` en su USB o en la nube y súbala a su repositorio de GitHub.

Próxima sesión

Práctica 2: estructuras de control y arreglos

LO QUE VIENE

- Decidir con `Si` y `Segun`.
- Repetir con `Mientras`, `Repetir` y `Para`.
- Guardar cinco notas en un **arreglo**.

PARA MAÑANA

1. Leer la Práctica 2 del manual.
2. Crear su cuenta de GitHub y el repositorio `Algoritmos-FE729`.
3. Traer la carpeta `Practica_01`: la usaremos de nuevo.

Esta tarde, en **Electrónica Digital**, veremos cómo la computadora evalúa `Y`, `O` y `NO` con compuertas lógicas reales.

UNIVERSIDAD RURAL DE GUATEMALA

¡Gracias!

¿Preguntas?

github.com/MynorCifuentes