



Facultad de Ingeniería Ingeniería y de Sistemas

# Algoritmia y Estructura de Datos: Introducción

# Revisión del Sílabo de Algoritmia y Estructura de Datos

- Desarrollo de algoritmos y aplicación de estructuras de datos para resolver problemas
  - Aplicación de un lenguaje de programación
    - Aplicación de estructuras de control secuenciales, selectivas y repetitivas
      - Operaciones con estructuras de datos estáticas: arreglos, registros, cadenas y sus combinaciones
      - Desarrollo de algoritmos de ordenamiento y búsqueda
    - Aplicación de subprogramas y recursividad
    - Concepto de puntero y operaciones con estructuras de datos dinámicas: listas, pilas, colas y árboles

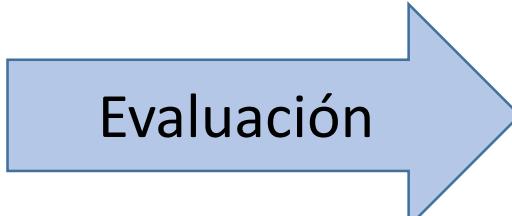
Sumilla

# Revisión del Sílabo de Algoritmia y Estructura de Datos

## Competencias

- 1 Desarrollar el razonamiento lógico para la solución de problemas
- 2 Seleccionar la estructura de datos más adecuada para representar los datos del problema y facilitar la lógica de solución
- 3 Integrar diversas estructuras de datos en la solución de un problema
- 4 Desarrollar la capacidad de análisis y diseño para implementar soluciones basadas en algoritmos.
- 5 Dominar un lenguaje de programación a nivel intermedio

# Revisión del Sílabo de Algoritmia y Estructura de Datos



Evaluación

- EP Examen Parcial (Peso 1)
- EF Examen Final (Peso 2)
- PC1, PC2, PC3 y PC4 Prácticas Calificadas
- PP Promedio de Prácticas
- PF Promedio Final

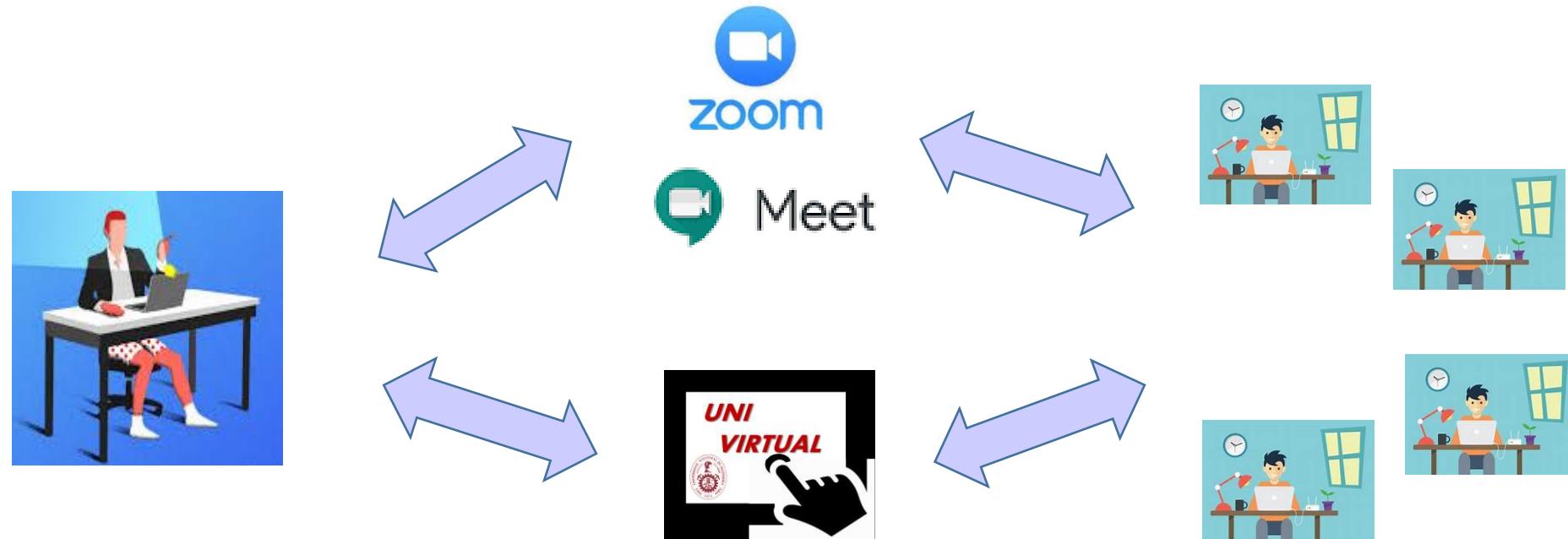
$$PP = (PC1+PC2+PC3+PC4-\text{m\'ınimo } (PC1..PC4))/3$$

$$PF = ( EP + EF * 2 + PP )/4$$

Nota: los días de prácticas calificadas y exámenes serán los martes de 8 a 10 am  
las prácticas calificadas y exámenes serán únicos para todas las secciones

## Modalidad no presencial

- Clases en línea, vía Zoom (martes 8-10 am), vía Google Meet (miércoles de 6-8 pm)
- Evaluaciones via UNIVirtual



# Razones por las que es importante la algoritmia en la formación del ingeniero

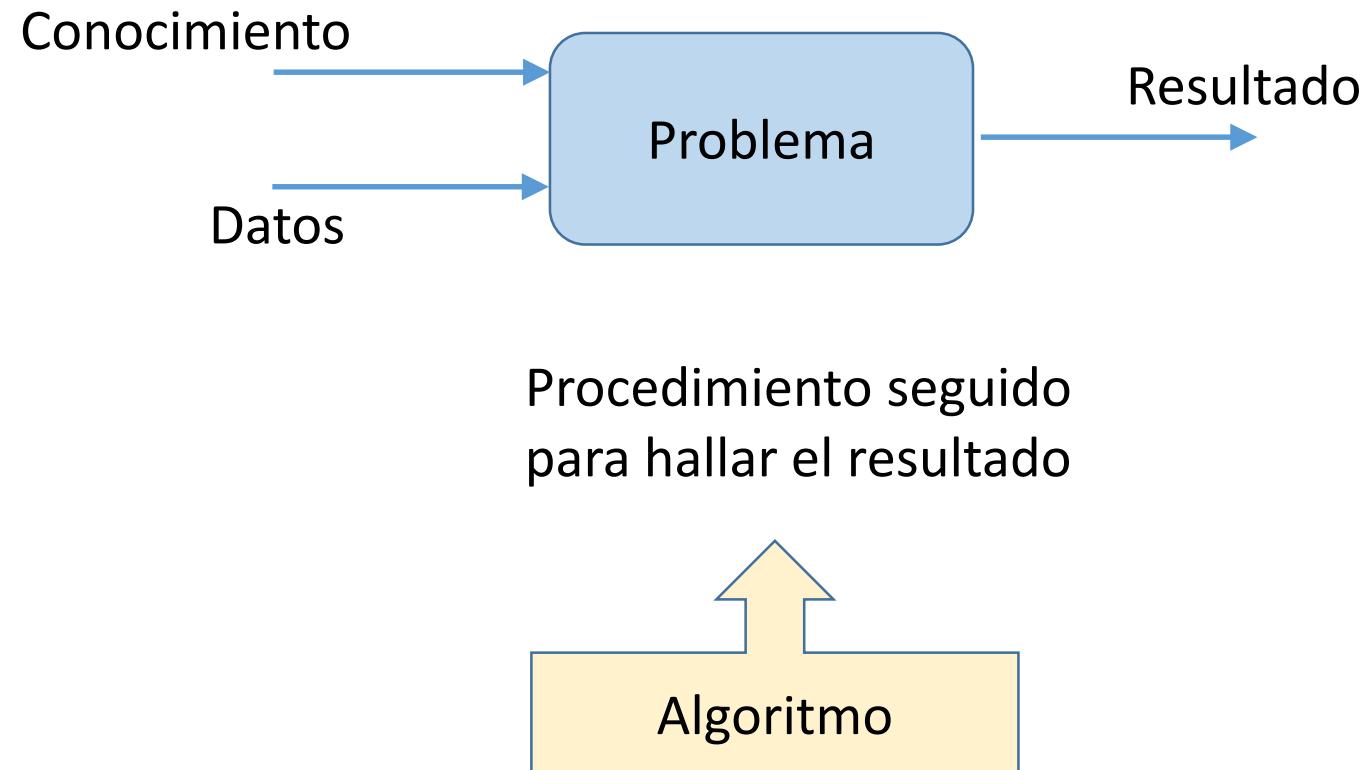
Primera

Desarrolla la capacidad de razonamiento lógico para la solución de problemas

Segunda

Desarrolla la capacidad de especificar de manera precisa

# Problema y Algoritmo



# Problema y Algoritmo

Problema

Hallar el Máximo Común Divisor de 2 números enteros positivos

Conocimiento

Regla 1:  $\text{MCD}(x, x) = x$

Regla 2: si  $x > y$  entonces  $\text{MCD}(x, y) = \text{MCD}(x-y, y)$   
si  $y > x$  entonces  $\text{MCD}(x, y) = \text{MCD}(x, y-x)$

Datos

Un par de números enteros positivos

Problema  
algorítmico

Especificar la secuencia de pasos que con el conocimiento dado permite hallar el resultado del problema

# Problema y Algoritmo

Algoritmo primera  
aproximación

```
declarar variables x,y como enteros  
  
leer x,y  
mientras x ≠ y  
    aplicar la regla 2  
finmientras  
escribir el resultado aplicando la regla 1
```

Algoritmo segunda  
aproximación

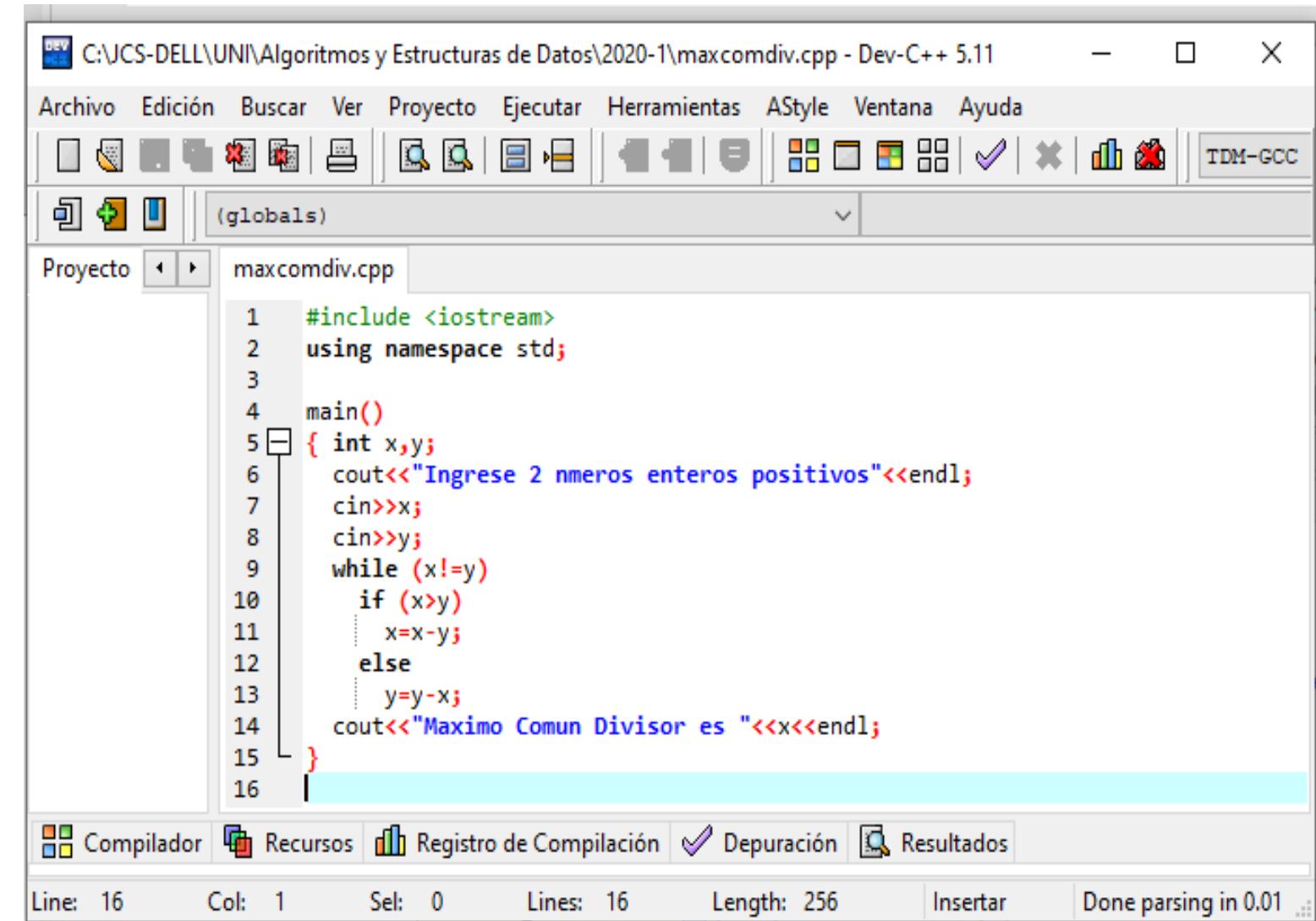
```
declarar variables x,y como enteros  
  
leer x,y  
mientras x ≠ y  
    si x > y entonces x = x-y  
    en caso contrario y = y-x  
finmientras  
escribir "Máximo Común Divisor es ", x
```

# Algoritmo y código en C++

## Algoritmo en pseudocódigo

```
declarar variables x,y como enteros  
  
leer x,y  
mientras x ≠ y  
    si x > y entonces x = x-y  
    en caso contrario y = y-x  
finmientras  
escribir "Máximo Común Divisor es ", x
```

## Código e C++



The screenshot shows the Dev-C++ IDE interface. The title bar reads "C:\JCS-DELL\UNI\Algoritmos y Estructuras de Datos\2020-1\maxcomdiv.cpp - Dev-C++ 5.11". The menu bar includes Archivo, Edición, Buscar, Ver, Proyecto, Ejecutar, Herramientas, AStyle, Ventana, and Ayuda. The toolbar has various icons for file operations like Open, Save, and Build. The project tree on the left shows "maxcomdiv.cpp" selected. The main code editor window displays the following C++ code:

```
1 #include <iostream>  
2 using namespace std;  
3  
4 main()  
5 { int x,y;  
6     cout<<"Ingrese 2 nmeros enteros positivos" << endl;  
7     cin>>x;  
8     cin>>y;  
9     while (x!=y)  
10     { if (x>y)  
11         x=x-y;  
12     else  
13         y=y-x;  
14     cout<<"Maximo Comun Divisor es " <<x<< endl;  
15 }  
16
```

The code uses standard input-output streams (`<iostream>`) and the `main()` function. It prompts the user to enter two positive integers. The program then uses a loop to calculate the greatest common divisor (GCD) by repeatedly subtracting the smaller number from the larger one until they are equal. Finally, it outputs the GCD.