

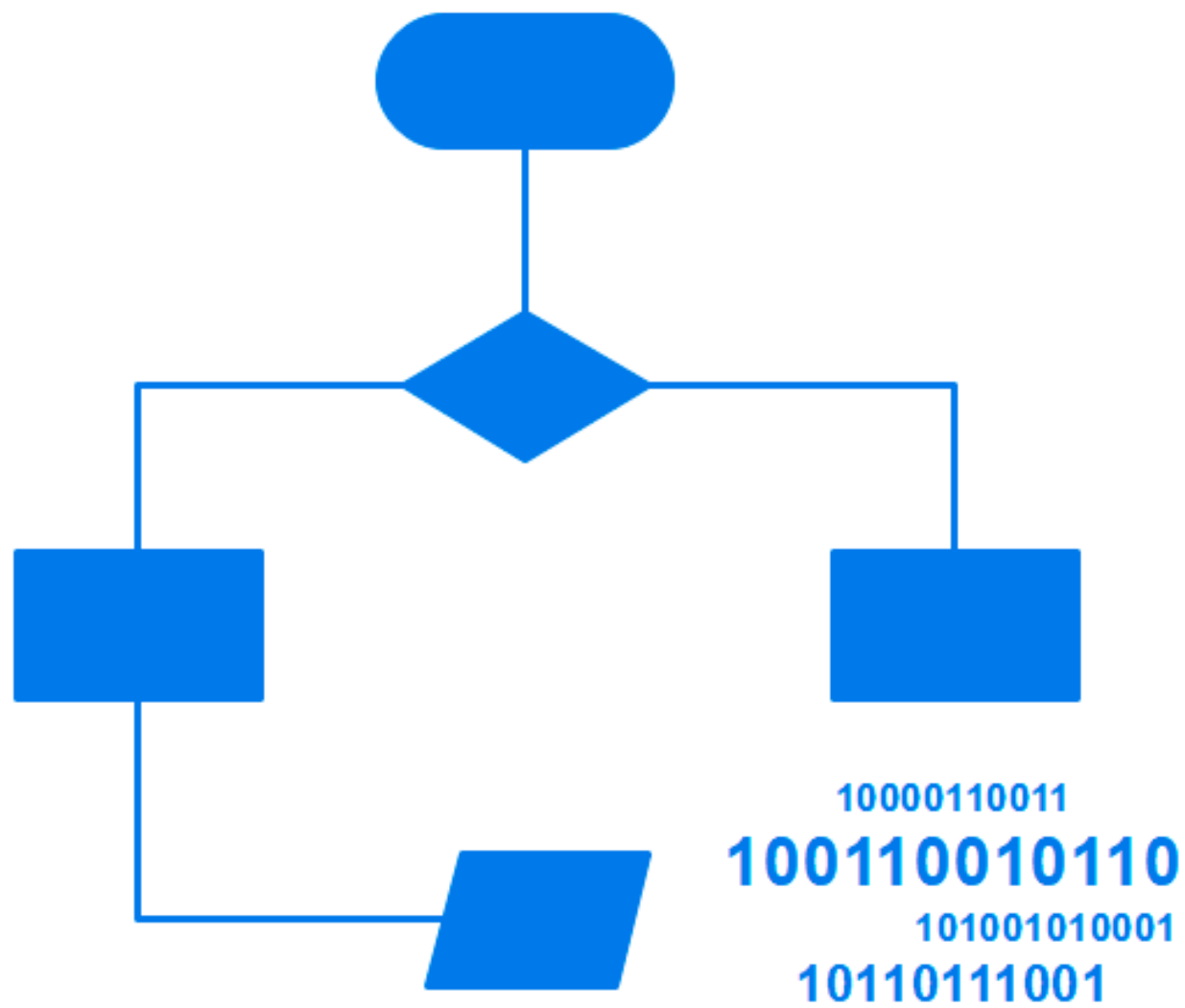
Explicación de Algoritmo y Diagrama de Flujo

El **algoritmo y el diagrama de flujo** son dos tipos de herramientas para explicar el proceso de un programa. En esta página, se discuten las diferencias entre un algoritmo y un diagrama de flujo al igual que la forma de crear un **diagrama de flujo** para ilustrar el algoritmo de forma visual.

Los algoritmos y los diagramas de flujo son dos herramientas diferentes que resultan útiles para crear nuevos programas, especialmente en la programación informática. Un algoritmo es un análisis paso a paso del proceso, mientras que un diagrama de flujo explica los pasos de un programa de forma gráfica.



Parte 1: Definición de algoritmo

Escribir un método lógico paso a paso para resolver un problema se denomina algoritmo. En otras palabras, un algoritmo es un procedimiento para resolver problemas. Es el primer paso del proceso para resolver un problema matemático o informático. Un algoritmo incluye cálculos, razonamientos y procesamiento de datos. Los algoritmos pueden presentarse mediante lenguajes naturales, pseudocódigo, diagramas de flujo, entre otros.



Parte 2: Definición de diagrama de flujo

Un diagrama de flujo es la representación gráfica de un algoritmo con la ayuda de diferentes símbolos, formas y flechas para demostrar un proceso o un programa, para así entenderlo fácilmente. El objetivo principal de utilizar un diagrama de flujo es analizar los diferentes métodos del mismo y para ello se aplican varios símbolos estándar:

Caja de Terminales - Inicio / Fin	
Entrada / Salida	
Proceso / Instrucción	
Decisión	
Conector / Flecha	

Los símbolos anteriores representan diferentes partes de un diagrama de flujo. El proceso en un diagrama de flujo puede expresarse a través de cuadros y flechas de diferentes tamaños y colores. En un diagrama de flujo, podemos destacar fácilmente ciertos elementos y las relaciones entre cada parte.

Parte 3: Diferencia entre algoritmo y diagrama de flujo

Parte 3: Diferencia entre algoritmo y diagrama de flujo

Si comparamos un diagrama de flujo con una película, un algoritmo es la historia de esa película. En otras palabras, **un algoritmo es el núcleo de un diagrama de flujo**. Actualmente, en el campo de la programación informática, existen muchas diferencias entre el algoritmo y el diagrama de flujo en varios aspectos, como la precisión, la forma en que se muestran y la forma en que la gente los percibe. A continuación, se muestra en detalle una tabla que ilustra las diferencias entre ambos.

Algoritmo	Diagrama de flujo
Es un procedimiento para resolver problemas.	Es una representación gráfica de un proceso.
El proceso se muestra con instrucciones paso a paso.	El proceso se muestra en un diagrama de información bloque por bloque.
Es complejo y difícil de entender.	Es intuitivo y fácil de entender.
Es conveniente depurar los errores.	Es difícil depurar los errores.
La solución se muestra en lenguaje natural.	La solución se presenta en formato gráfico.
Es un poco más fácil resolver un problema complejo.	Es difícil resolver un problema complejo.
Cuesta más tiempo crear un algoritmo.	Crear un diagrama de flujo cuesta menos tiempo.

Parte 4: Tipos de algoritmos

No es de extrañar que los algoritmos se utilicen ampliamente en la programación informática. Sin embargo, se pueden aplicar para resolver problemas matemáticos e incluso en la vida cotidiana. Aquí surge una pregunta: ¿cuántos tipos de algoritmos existen? Según el Dr. Christoph Koutschan, informático que trabaja en el Instituto de Investigación para la Computación Simbólica (RISC) de Austria, ha realizado una encuesta sobre los tipos de algoritmos más importantes. Como resultado, ha elaborado una lista de 32 algoritmos cruciales en informática. A pesar de la complejidad de los algoritmos, en general podemos dividirlos en seis tipos fundamentales basados en su función.

1. Algoritmo recursivo

Se refiere a una forma de resolver problemas dividiendo repetidamente el problema en subproblemas del mismo tipo. El ejemplo clásico del uso de un algoritmo recursivo para resolver problemas es la Torre de Hanoi.

2. Algoritmo de divide y vencerás

Tradicionalmente, el algoritmo "divide y vencerás" consta de dos partes:

1. descomponer un problema en algunos subproblemas independientes más pequeños del mismo tipo; 2. encontrar la solución final de los problemas originales después de resolver estos problemas más pequeños por separado.

Los puntos clave del algoritmo "divide y vencerás" son:

Si puedes encontrar los subproblemas repetidos y la subestructura del bucle del problema original, puedes convertir rápidamente el problema original en una cuestión pequeña y sencilla.

Intenta desglosar toda la solución en varios pasos (diferentes pasos necesitan diferentes soluciones) para facilitar el proceso.

¿Son los subproblemas fáciles de resolver? Si no fuera así, el problema original podría costar mucho tiempo.

3. Algoritmo de programación dinámica

Desarrollado por Richard Bellman en los años 50, el algoritmo de programación dinámica se utiliza generalmente para problemas de optimización. En este tipo de algoritmo, se recogen los resultados pasados para utilizarlos en el futuro. Al igual que el algoritmo "divide y vencerás", un algoritmo de programación dinámica simplifica un problema complejo dividiéndolo en algunos subproblemas sencillos. Sin embargo, la diferencia más significativa entre ellos es que el segundo requiere la superposición de subproblemas, mientras que el primero no lo necesita.

4. Algoritmo codicioso

Esta se refiere a encontrar siempre la mejor solución en cada paso en lugar de considerar la optimización general. Debido a las limitaciones del algoritmo codicioso, debe tenerse en cuenta que la clave para elegir un algoritmo codicioso es considerar las consecuencias en el futuro.

5. Algoritmo de fuerza bruta

El algoritmo de fuerza bruta es una solución simple y directa del problema, generalmente basada en la descripción del problema y la definición del concepto involucrado. También se puede utilizar "¡solo hazlo!" para describir la estrategia de fuerza bruta. En resumen, un algoritmo de fuerza bruta se considera uno de los algoritmos más simples, que itera todas las posibilidades y termina con una solución satisfactoria.

6. Algoritmo de retroceso

Basado en una búsqueda recursiva en profundidad, el algoritmo de backtracking se centra en encontrar la solución al problema durante el proceso de búsqueda tipo enumeración. Cuando no se pueda satisfacer la condición, devolverá el "backtracking" e intentará otro camino. Es adecuado para resolver problemas grandes y complicados, por lo que se ha ganado la reputación de "método de solución general". Uno de los ejemplos más famosos de algoritmos de backtracking es el rompecabezas de las ocho reinas.

Parte 5: Utilizar diagramas de flujo para representar algoritmos

Ahora que hemos aprendido las definiciones de algoritmo y diagrama de flujo, ¿cómo podemos utilizar un **diagrama de flujo** para representar un algoritmo? Para crear un diagrama de flujo de un algoritmo, necesitamos utilizar una herramienta de diagramación práctica como EdrawMax para terminar el trabajo.

Los algoritmos se utilizan principalmente para programas matemáticos e informáticos, mientras que los **diagramas de flujo** pueden utilizarse para describir todo tipo de procesos: empresariales, educativos, personales y algoritmos. Por eso, los diagramas de flujo se utilizan a menudo como herramienta de planificación de programas para organizar visualmente el proceso paso a paso del programa. A continuación, algunos ejemplos:

Ejemplo 1: Imprime del 1 al 20:

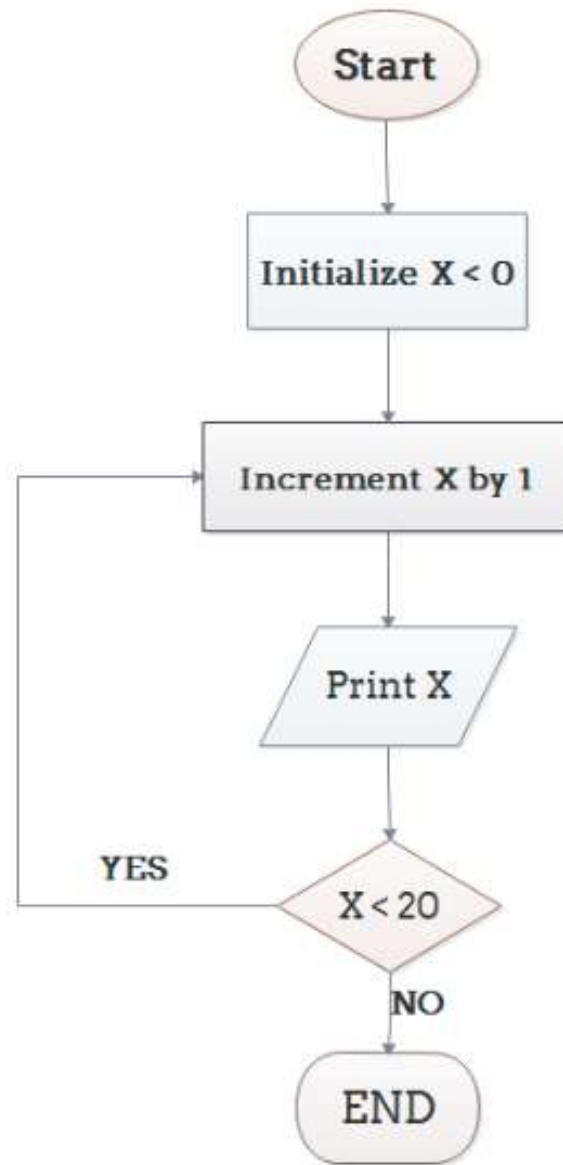
Algoritmo:

Paso 1: Inicializar X como 0,

Paso 2: Incrementa X en 1,

Paso 3: Imprimir X,

Paso 4: Si X es inferior a 20, vuelva al paso 2.



Ejemplo 2: Convertir Temperatura de Fahrenheit (°F) a Celsius (°C)

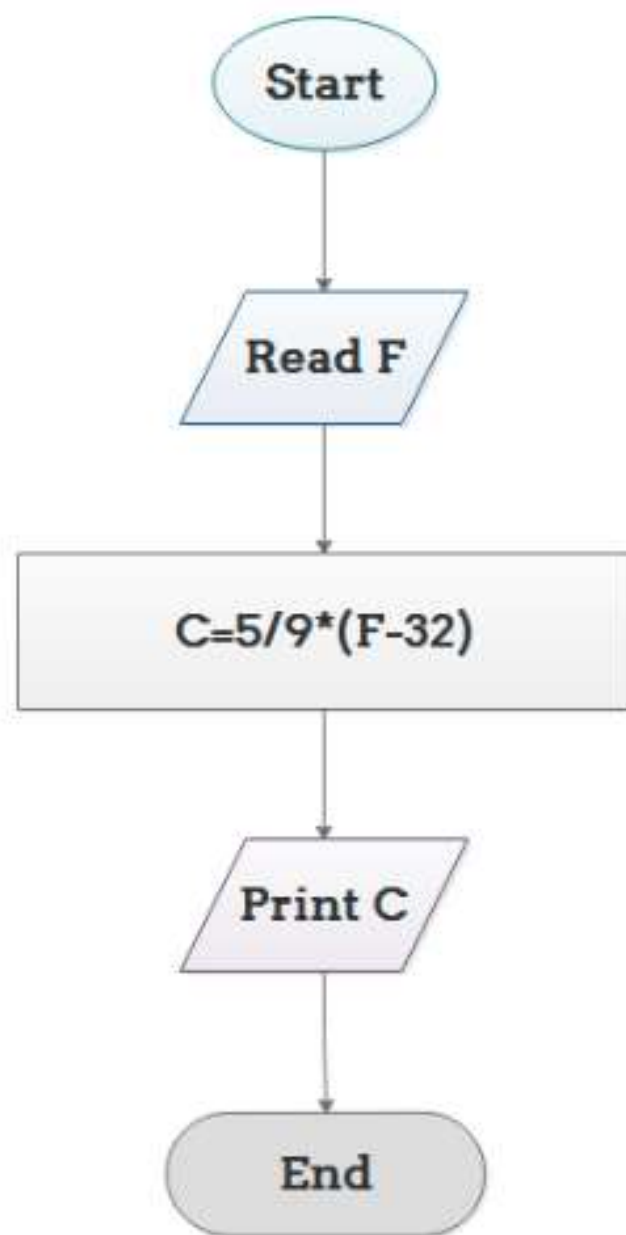
Algoritmo:

Paso 1: Leer la temperatura en Fahrenheit,

Paso 2: Calcular la temperatura con la fórmula

$C = 5/9 * (F - 32)$,

Paso 3: Imprimir C.



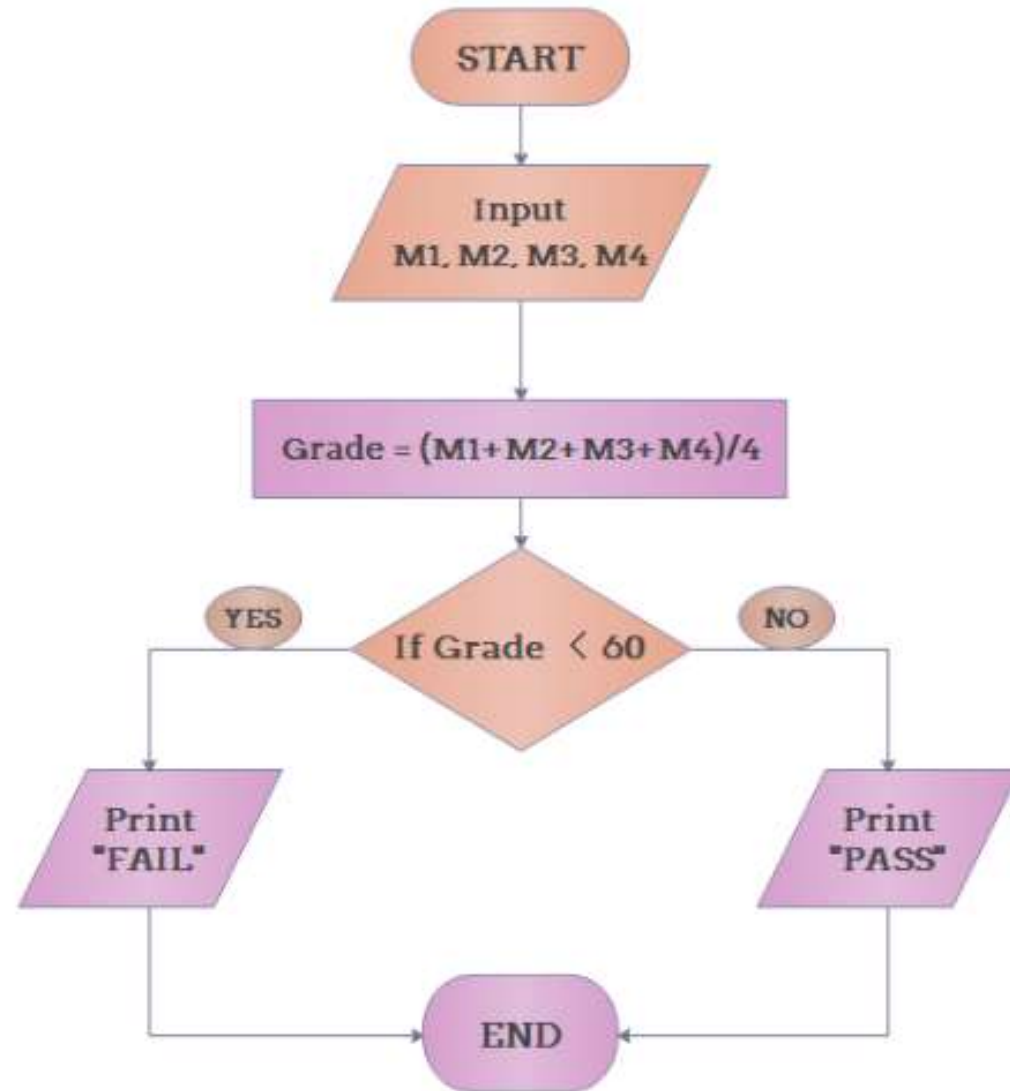
Ejemplo 3: Determinar si un alumno ha aprobado el examen o no:

Algoritmo:

Paso 1: Notas de entrada de 4 cursos M1, M2, M3 y M4,

Paso 2: Calcular la nota media con la fórmula
$$\text{"Nota"} = (M1 + M2 + M3 + M4) / 4$$

Paso 3: Si la nota media es inferior a 60, imprime "FALLO", si no imprime "APROBADO".



Parte 6: Conclusión

De lo anterior, podemos llegar a la conclusión de que un **diagrama de flujo** es una representación gráfica de un algoritmo, un algoritmo puede ser expresado y analizado a través de un diagrama de flujo. Un algoritmo muestra cada paso para llegar a la solución final, mientras que un diagrama de flujo muestra cómo llevar a cabo el proceso conectando cada paso. Un algoritmo utiliza principalmente palabras para describir los pasos, mientras que se puede crear un diagrama de flujo con símbolos de diagrama de flujo para hacer el proceso más lógico.

Ejercicios

Desarrolle un algoritmo que permita convertir calificaciones numéricas, según la siguiente tabla: A = 19 y 20, B = 16, 17 y 18, C = 13, 14 y 15, D = 10, 11 y 12, E = 1 hasta el 9. Se asume que la nota está comprendida entre 1 y 20.

Guía Completa: Algoritmos y Diagramas de Flujo desde Cero

Introducción

El algoritmo y el diagrama de flujo son dos herramientas esenciales para explicar el proceso de un programa. Los algoritmos son instrucciones paso a paso para resolver un problema, mientras que los diagramas de flujo representan gráficamente estos pasos utilizando símbolos estandarizados. ¿Qué es un Algoritmo?

Un algoritmo es una secuencia lógica y finita de pasos que permiten resolver un problema. Puede representarse de diversas formas, como en lenguaje natural, pseudocódigo o diagramas de flujo.

Diferencias entre Algoritmo y Diagrama de Flujo

Si comparamos un diagrama de flujo con una película, un algoritmo es la historia de esa película. Mientras que un algoritmo es una serie de instrucciones paso a paso, el diagrama de flujo muestra cómo se ejecuta visualmente el proceso.

Taller de Diagramas de Flujo

En esta sección, aprenderás a construir diagramas de flujo a partir de problemas sencillos.

Creación de Diagramas de Flujo

Ejercicio 1: Saludo al usuario

Crear un diagrama de flujo que solicite el nombre de un usuario y lo salude.

Ejercicio 2: Calcular la edad

Solicitar el año de nacimiento del usuario y calcular su edad actual.

Ejercicio 3: Conversión de temperatura

Convertir grados Celsius a Fahrenheit.

Ejercicio 4: Suma de dos números

Solicitar dos números y mostrar su suma.

Ejercicio 5: Multiplicación de dos números

Solicitar dos números y mostrar su producto.

Módulo 2: Decisiones en Diagramas de Flujo

Ejercicio 6: Número par o impar

Determinar si un número ingresado por el usuario es par o impar.

Ejercicio 7: Mayor de edad

Determinar si un usuario es mayor o menor de edad.

Ejercicio 8: Comparar dos números

Comparar dos números y mostrar cuál es mayor.

Ejercicio 9: Verificar si un número es positivo o negativo

Indicar si un número ingresado es positivo, negativo o cero.

Ejercicio 10: Aprobar o reprobar un examen

Determinar si un estudiante aprobó un examen según su calificación.

Conclusión

Los algoritmos y diagramas de flujo son herramientas fundamentales en la resolución de problemas en programación. Es recomendable practicar la creación de diagramas de flujo para mejorar la lógica antes de escribir código.

Ejercicios Resueltos Paso a Paso

A continuación, se presentan cinco ejercicios resueltos con explicaciones detalladas paso a paso para que puedas comprender cómo aplicar algoritmos y diagramas de flujo en situaciones cotidianas.

Ejercicio 1: Saludar a un Usuario

Problema: Crear un algoritmo que solicite el nombre del usuario y lo salude.

Explicación: Para este ejercicio, el sistema debe pedir un nombre y luego mostrar un mensaje de saludo.

Pasos:

1. Iniciar el proceso.
2. Pedir al usuario que ingrese su nombre.
3. Leer el nombre ingresado.
4. Mostrar un mensaje de saludo con el nombre ingresado.
5. Finalizar el proceso.

1. **Óvalo:** Escribe "Inicio".
2. **Flecha:** Conecta el óvalo a un **paralelogramo** que diga "Pedir nombre".
3. **Flecha:** Conecta el paralelogramo a un **rectángulo** que diga "Leer nombre".
4. **Flecha:** Conecta el rectángulo a otro **rectángulo** que diga "Mostrar saludo".
5. **Flecha:** Conecta el último rectángulo a un **óvalo** que diga "Fin".

Herramientas para Dibujar los Diagramas

- **Lucidchart:** Herramienta en línea fácil de usar.
- **Draw.io:** Gratuita y muy intuitiva.
- **Microsoft Visio:** Profesional y avanzada.
- **PowerPoint:** Puedes usar formas y conectores para crear diagramas.

Ejercicio 2: Determinar si un Número es Par o Impar

Problema: Crear un algoritmo que determine si un número ingresado por el usuario es par o impar.

Explicación: Un número es par si al dividirlo entre 2 su residuo es 0, de lo contrario es impar.

Pasos:

1. Iniciar el proceso.
2. Pedir al usuario que ingrese un número.
3. Leer el número ingresado.
4. Verificar si el número es divisible por 2.
 - Si el residuo es 0, mostrar 'El número es par'.
 - Si el residuo no es 0, mostrar 'El número es impar'.
5. Finalizar el proceso.

1. **Óvalo:** Escribe "Inicio".
2. **Flecha:** Conecta el óvalo a un **paralelogramo** que diga "Pedir número".
3. **Flecha:** Conecta el paralelogramo a un **rectángulo** que diga "Leer número".
4. **Flecha:** Conecta el rectángulo a un **rombo** que diga "¿Número divisible por 2?".
 - **Flecha "Sí":** Conecta a un **rectángulo** que diga "Mostrar 'Es par'".
 - **Flecha "No":** Conecta a un **rectángulo** que diga "Mostrar 'Es impar'".
5. **Flecha:** Conecta ambos rectángulos a un **óvalo** que diga "Fin".

Ejercicio 3: Conversión de Temperatura

Problema: Crear un algoritmo que convierta grados Celsius a Fahrenheit.

Explicación: Para convertir grados Celsius a Fahrenheit usamos la fórmula: $F = (C \times 9/5) + 32$.

Pasos:

1. Iniciar el proceso.
2. Pedir al usuario que ingrese la temperatura en grados Celsius.
3. Leer la temperatura ingresada.
4. Aplicar la fórmula $F = (C \times 9/5) + 32$.
5. Mostrar el resultado en grados Fahrenheit.
6. Finalizar el proceso.

1. **Óvalo:** Escribe "Inicio".
2. **Flecha:** Conecta el óvalo a un **paralelogramo** que diga "Pedir temperatura en Celsius".
3. **Flecha:** Conecta el paralelogramo a un **rectángulo** que diga "Leer temperatura".
4. **Flecha:** Conecta el rectángulo a otro **rectángulo** que diga "Aplicar fórmula $F = (C \times 9/5) + 32$ ".
5. **Flecha:** Conecta el rectángulo a otro **rectángulo** que diga "Mostrar resultado en Fahrenheit".
6. **Flecha:** Conecta el último rectángulo a un **óvalo** que diga "Fin".

Ejercicio 4: Calcular el Área de un Triángulo

Problema: Crear un algoritmo que calcule el área de un triángulo dado su base y altura.

Explicación: La fórmula del área del triángulo es: $\text{Área} = (\text{Base} \times \text{Altura}) / 2$.

Pasos:

1. Iniciar el proceso.
2. Pedir al usuario que ingrese la base del triángulo.
3. Pedir al usuario que ingrese la altura del triángulo.
4. Leer la base y la altura ingresadas.
5. Aplicar la fórmula del área.

6. Mostrar el resultado del área del triángulo.

7. Finalizar el proceso.

1. **Óvalo:** Escribe "Inicio".
2. **Flecha:** Conecta el óvalo a un **paralelogramo** que diga "Pedir base".
3. **Flecha:** Conecta el paralelogramo a otro **paralelogramo** que diga "Pedir altura".
4. **Flecha:** Conecta el paralelogramo a un **rectángulo** que diga "Leer base y altura".
5. **Flecha:** Conecta el rectángulo a otro **rectángulo** que diga "Aplicar fórmula $\text{Área} = (\text{Base} \times \text{Altura}) / 2$ ".
6. **Flecha:** Conecta el rectángulo a otro **rectángulo** que diga "Mostrar área".
7. **Flecha:** Conecta el último rectángulo a un **óvalo** que diga "Fin".

Ejercicio 5: Verificar si un Año es Bisiesto

Problema: Crear un algoritmo que determine si un año ingresado por el usuario es bisiesto.

Explicación: Un año es bisiesto si es divisible por 4, pero no por 100, salvo que también sea divisible por 400.

Pasos:

1. Iniciar el proceso.
2. Pedir al usuario que ingrese un año.
3. Leer el año ingresado.
4. Aplicar las siguientes reglas:
 - Si el año es divisible por 400, es bisiesto.
 - Si el año es divisible por 100 pero no por 400, no es bisiesto.
 - Si el año es divisible por 4 pero no por 100, es bisiesto.
 - En cualquier otro caso, no es bisiesto.
5. Mostrar el resultado.
6. Finalizar el proceso.

1. **Óvalo**: Escribe "Inicio".
2. **Flecha**: Conecta el óvalo a un **paralelogramo** que diga "Pedir año".
3. **Flecha**: Conecta el paralelogramo a un **rectángulo** que diga "Leer año".
4. **Flecha**: Conecta el rectángulo a un **rombo** que diga "¿Año divisible por 400?".
 - **Flecha "Sí"**: Conecta a un **rectángulo** que diga "Mostrar 'Es bisiesto'".
 - **Flecha "No"**: Conecta a otro **rombo** que diga "¿Año divisible por 100?".
 - **Flecha "Sí"**: Conecta a un **rectángulo** que diga "Mostrar 'No es bisiesto'".
 - **Flecha "No"**: Conecta a otro **rombo** que diga "¿Año divisible por 4?".
 - **Flecha "Sí"**: Conecta a un **rectángulo** que diga "Mostrar 'Es bisiesto'".
 - **Flecha "No"**: Conecta a un **rectángulo** que diga "Mostrar 'No es bisiesto'".
5. **Flecha**: Conecta todos los rectángulos a un **óvalo** que diga "Fin".