# Programación en C++

# Resolución de Problemas, Algoritmos y Programas

# Conceptos Básicos

*Algoritmo: Técnica de resolución de problemas mediante el cual se elabora una secuencia razonada de pasos para encontrar una solución.*

*Lenguaje de programación: Conjunto de instrucciones (órdenes) y reglas que conforman un lenguaje que la computadora entiende.*

*Programas: Es la representación de un algoritmo usando un lenguaje de programación determinado.*

# Lenguajes de Programación

¿Cómo decirle a la computadora que resuelva un problema?

- *Lenguajes de máquina*
- *Lenguajes de ensamblador (de bajo nivel)*
- *Lenguajes de alto nivel*

*La computadora solo puede entender directamente su propio lenguaje de máquina*

*Para las personas es muy difícil entender el lenguaje de máquina, ¿qué hacer?*

# Traductores de Lenguajes

*Un lenguaje de alto nivel es como un idioma extranjero para la máquina … necesitamos de los traductores.*

***Intérpretes**: traduce y a continuación lo ejecuta.*

***Compiladores**: traduce, guarda lo traducido en un archivo para su posterior ejecución.*

Usamos la abstracción para luchar con la complejidad del lenguaje de máquina, pero … nada es gratuito!!

# Resolución de Problemas

- *Tema central de la computación.*
- *La programación es una forma especializada de resolución de problemas.*

*Las fases de resolución de un problema con computadora son:*

- Análisis del problema
- Diseño del algoritmo
- Codificación
- Compilación y ejecución
- Verificación
- Depuración
- Mantenimiento
- Documentación

# Algoritmo

- Es un método para resolver un problema a través de una secuencia de pasos que llevará a cumplir un objetivo o una solución, en un tiempo finito.

*Ejemplos*

- El uso de una lavadora automática.
- El algoritmo para la conversión de Decimal a Binario.
- El algoritmo para resolver un sistema de ecuaciones lineales.

# Características de los Algoritmos

1. Carácter Finito: Un algoritmo siempre debe terminar en un tiempo razonable.
2. Precisión: Cada paso del algoritmo debe ser claramente definido, sin ambigüedades.
3. Entrada: Tiene algunos valores que puede necesitar el algoritmo para empezar.
4. Salida: Es el producto del algoritmo.
5. Eficacia: Las operaciones deben ser básicas para que puedan ser hechas de manera exacta.

# Ejemplo: algoritmo para cambiar una bombilla o foco

1. Retirar la bombilla fundida
    1.1 Colocar una escalera debajo de la bombilla.
    1.2. Subir por la escalera.
    1.3. Desenroscar la bombilla en el sentido contrario de las agujas del reloj.
    1.4. Bajar la escalera.
2. Poner una bombilla nueva
    2.1. Coger la bombilla nueva
    2.2. Subir por la escalera
    2.3. Enroscar la bombilla en el sentido de las agujas del reloj
    2.4. Bajar la escalera
    2.5. Retirar la escalera

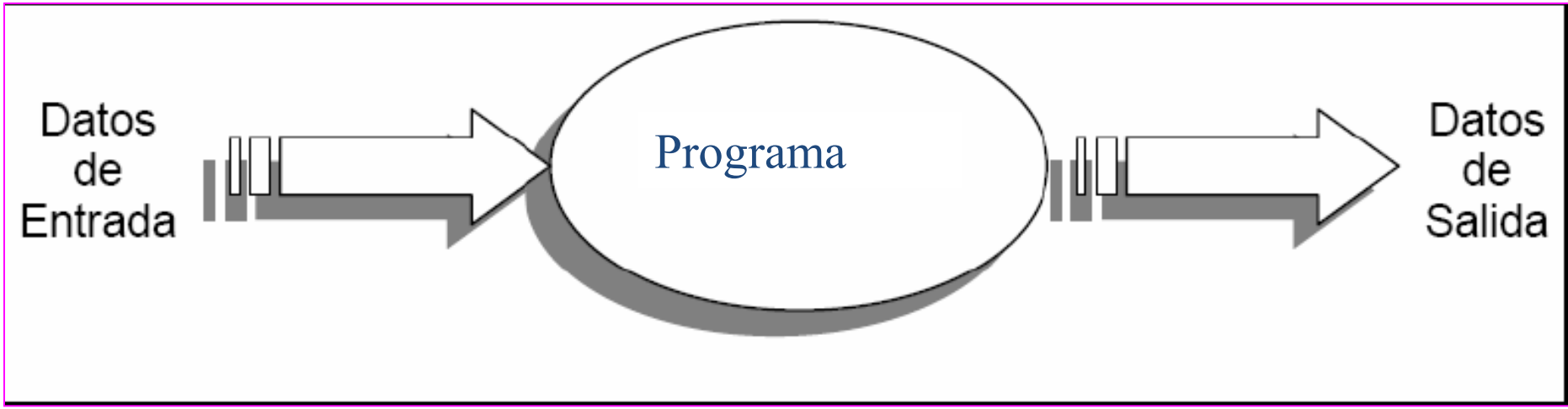
# Ejemplo: una receta de cocina no es un algoritmo!?

Por que:

- NO es repetible, pues para las mismas entradas (ingredientes), no se garantizan los mismos resultados.
- NO es preciso, ya que en una receta concreta no se suelen especificar los grados de temperaturas exactos, tipos de recipientes, calidad o tipo de ingredientes, etc.

# Programa

Un programa describe una transformación de unos datos de entrada para obtener unos datos de salida mediante el procesamiento de los datos.



# En Resumen

- Los algoritmos son esenciales en la programación.
- La programación es una de las actividades más importantes en la informática.
- Las computadoras hacen lo que nosotros le decimos que haga NO lo que quisiéramos que ellas hagan.
- Programación de computadoras implica escribir instrucciones y llevarlas a la computadora para que realicen una tarea.
- Aprenderemos a resolver problemas concretos mediante técnicas que se estudiarán en los temas siguientes.

# Ejemplo: Calcular el área de un rectángulo

Lo primero es plantearse las siguientes preguntas:

- **Datos de Entrada:**

  ¿Qué datos son de entrada?

  ¿Cuántos son los datos?

- **Salida:**

¿ Cuales y cuantos son los datos de salida? ¿Qué forma tiene la salida?

- Luego se definen los **procesos** para convertir la entradas en salidas.

# Diseño del algoritmo

Especificaciones de Entrada:

- Base
- Altura

Especificaciones de Salida:

- Resultado del área

Proceso:

- Area = Base * Altura

# Verificación:

- Una vez terminado de escribir un algoritmo es necesario comprobar que realiza las tareas para las que se ha diseñado y produce el resultado correcto y esperado.
- Se puede hacer mediante un ejercicio manual, escribiendo datos de entrada y haciendo los cálculos paso a paso, siguiendo la ruta del algoritmo.

# Implementación del algoritmo

- Una vez que el algoritmo está diseñado, representado mediante una técnica (diagrama de flujo o seudocódigo), y verificado:

- Se pasa a la codificación en un lenguaje de programación.

# Representación de Algoritmos

- Durante la fase de diseño, es preciso disponer de alguna notación algorítmica que permita expresar en forma no ambigua la resolución del problema.
- Las notaciones algorítmicas deben ser independientes del lenguaje de programación a utilizar en la fase de implementación, pero deben permitir una traducción clara.
- Tipos:
  - Gráfica: Diagramas de Flujo
  - Textual: Pseudocódigo

# Diagramas de Flujo

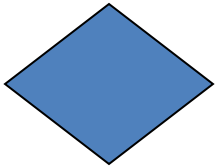
- Un diagrama de flujo es una herramienta gráfica para la representación de algoritmos.
- Consta de una serie de símbolos conectados mediante líneas que indican el orden en que se realizan las acciones.
- Debe reflejar la lógica del algoritmo indicando los pasos individuales y sus conexiones.
  - Muestra el comienzo del programa.
  - Las operaciones y el orden que se realizan.
  - El final del programa.

# Símbolos de Diagramas de Flujo

- Inicio y fin de programa

- Proceso

- Datos (entrada y salida)



- Decisión



- Control de flujo

- Conector

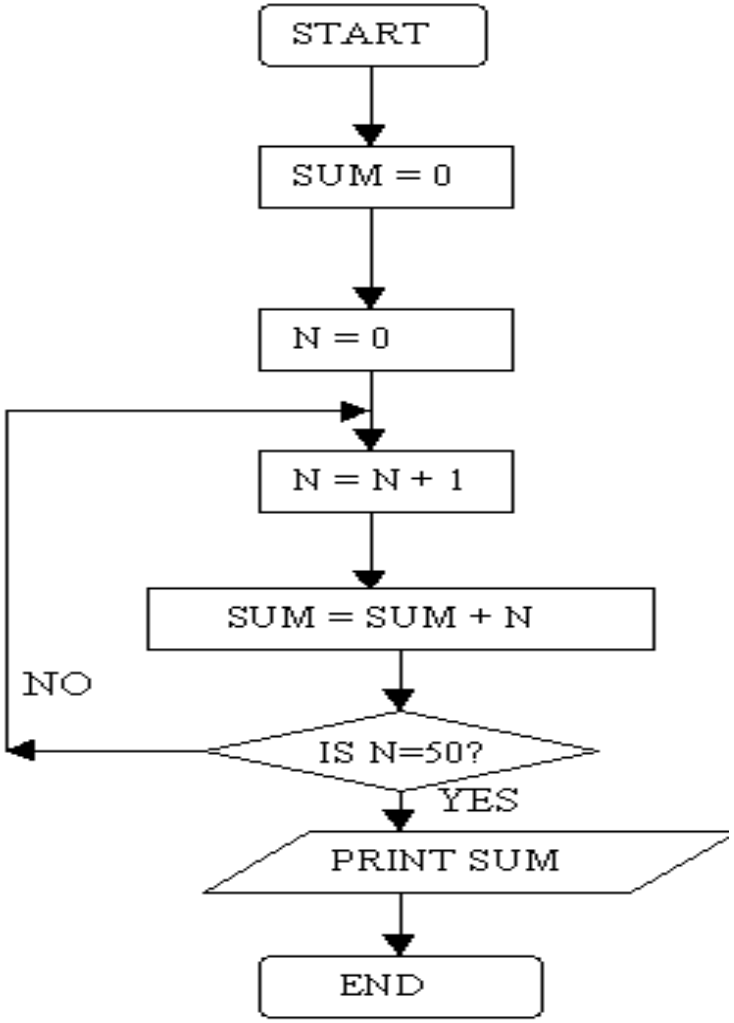
# Algunas reglas:

- El inicio del programa figurará en la parte superior del diagrama.
- Los símbolos de comienzo y fin deberán aparecer una única vez.
- El flujo de operaciones es de arriba hacia abajo y de izquierda a derecha.
- Se debe guardar cierta simetría en la representación de bucles y bifurcaciones.
- Se evitarán los cruces de líneas de flujo, utilizando conectores.

# Más Reglas

- Los símbolos se unen con líneas, las cuales tienen en la punta una flecha que indica la dirección que fluye la información, se deben de utilizar solamente líneas de flujo horizontal o verticales (nunca diagonales).
- Se debe evitar el cruce de líneas, para lo cual si quisiera separar el flujo del diagrama a un sitio distinto, se puede realizar utilizando los conectores. Usar conectores cuando sea estrictamente necesario.
- No deben quedar líneas de flujo sin conectar.
- Todo texto escrito dentro de un símbolo debe ser legible, preciso, evitando el uso de muchas palabras.
- Todos los símbolos pueden tener más de una línea de entrada, a excepción del símbolo final.
- Solo los símbolos de decisión pueden y deben tener mas de una línea de flujo de salida.

**Ejemplo:** Diagrama para encontrar la suma de los primeros 50 números naturales.



# Pseudocódigo

- El pseudocódigo describe un algoritmo utilizando una mezcla de frases en lenguaje común, instrucciones de programación y palabras clave que definen las estructuras básicas.
- Su objetivo es permitir que el programador se centre en los aspectos lógicos de la solución de un problema.

# Ventajas de utilizar un Pseudocódigo a un diagrama de flujo

- Permite representar de forma fácil operaciones repetitivas complejas.
- Es más sencilla la tarea de pasar de pseudocódigo a un lenguaje de programación formal.
- Si se usa sangría apropiadamente se puede observar claramente los niveles en la estructura del programa.
- En los procesos de aprendizaje de programación, los pseudocódigos están más cerca del paso siguiente, la codificación en un lenguaje de programación, con respecto a los diagramas de flujos.
- En el curso se desarrollará ambos: diagramas de flujo y pseudocódigo.

# Ejemplos:

Algoritmo que muestre cómo usted iría al cine.

Análisis del Problema:

**Datos de Salida:** Ver la película

**Datos de entrada:** Nombre de la película, dirección del cine, hora de proyección.

**Datos Auxiliares**: Entrada, número de asiento

**Proceso**: Se debe seleccionar una película de la cartelera del periódico, ir a la sala y comprar la entrada para finalmente ver la película.

# Diseño del algoritmo:

inicio
//seleccionar la película

tomar el periódico
mientras no lleguemos a la cartelera
    pasar la hoja
mientras no se acabe la cartelera
    leer película
si nos gusta, recordarla
elegir una de las películas seleccionadas
leer la dirección de la sala y la hora de proyección

//comprar la entrada
trasladarse a la sala
si no hay entradas, ir al fin
si hay cola
    ponerse al último
    mientras no lleguemos a la taquilla
    avanzar
    si no hay entradas, ir a fin
comprar la entrada
//ver la película
leer número de asiento de la entrada
buscar al asiento
sentarse
ver película
fin

Realice un algoritmo que ejemplifique como usted PONE LA MESA PARA LA COMIDA.

## Análisis del Problema:

- **Datos de Salida**: La mesa puesta
- **Datos de Entrada**: La vajilla, los cubiertos, las servilletas, el número de comensales.
- **Datos Auxiliares**: Número de platos, vasos, cubiertos o servilletas que llevamos puestos.
- **Proceso**: Para poner la mesa, después de poner el mantel, se toman las servilletas hasta que su número coincida con el de comensales y se colocan. La operación se repite para los vasos, platos y cubiertos.

# Diseño del algoritmo:

inicio
poner el mantel
repetir
    tomar una servilleta
hasta que el número de servilletas es igual al de comensales
repetir
    tomar un vaso
hasta que el número de vasos es igual al de comensales
repetir
    tomar un juego de platos
hasta que el número de juegos es igual al de comensales
repetir
    tomar un juego de cubiertos
hasta que el número de juegos es igual al de comensales
Fin

Ejemplo: Escriba un algoritmo para determinar la nota final de los alumnos e indicar si pasa o no. La nota final se calcula como el promedio de cuatro notas.

**Pseudocódigo**:

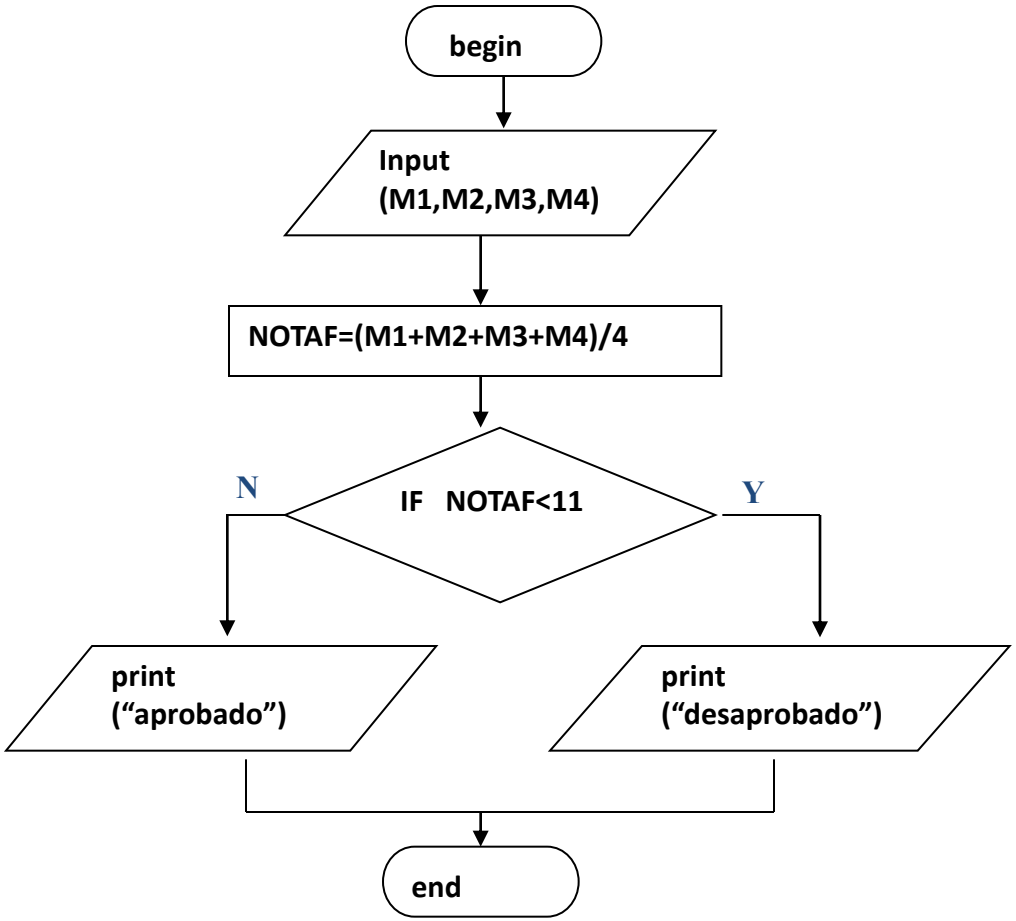
*Ingresar un conjunto de 4 notas*
*Calcular su promedio sumando y dividiendo por 4*
*Si el promedio es menor que 11*
*Imprimir "desaprobado"*
*De lo contrario*
*Imprimir "aprobado"*

**Algoritmo detallado**:

```
Paso 1:   input (M1, M2, M3, M4)
Paso 2:   NOTAF = (M1+M2+M3+M4)/4
Paso 3:   if (NOTAF < 11) then
                    print ( "desaprobado")
          else
                    print ("aprobado")
```

**Ejemplo:** Diseña un diagrama de flujo para determinar la nota final de los alumnos e indicar si pasa o no. La nota final se calcula como el promedio de cuatro notas.

## Diagrama de Flujo




**VERIFICACIÓN:**

| NOTA | NOTAF | STATUS |
|---|---|---|
| M1= | 15 | 15 |
| M2= | 7 | + 07 |
| M3= | 8 | + 08 |
| M4= | 10 | + 10 /4 |
| | = | 10 |

**DESAPROBADO**