Webinar #24

Los lenguajes de programación en la ingeniería: del aprendizaje a la aplicación

Francisco García Herrera

Profesor–Investigador del Departamento de Irrigación

Universidad Autónoma Chapingo (UACh)

16 de julio del 2020

Contenido

1. Metodología de la Programación

1.1. El Almacenamiento de la información y las variables
1.2. Pasos básicos para programar
1.3. El código fuente
1.4. Puesta a punto de un programa
1.5. Los errores clásicos al programar

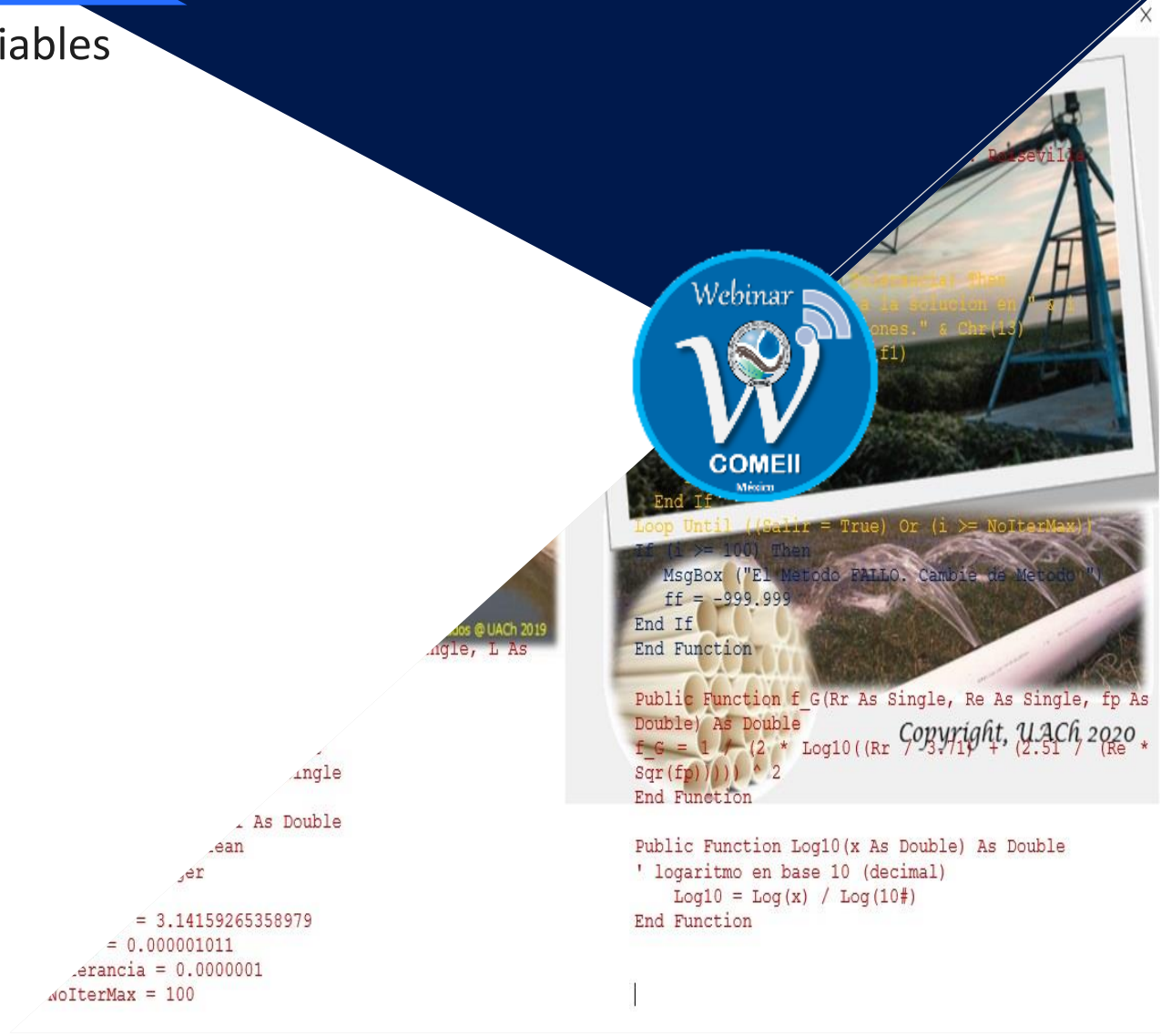
2. Los lenguajes de programación

2.1. Definición y tipos de lenguajes
2.2. La Consola
2.3. La GUI
2.4. La Consola vs la GUI

3. Desarrollo de aplicaciones

3.1. Elección de la plataforma de desarrollo
3.2. La facilidad de usar un RAD
3.3. Tipo de aplicaciones en la ingeniería
3.4. La utilidad de las aplicaciones a la medida.

4. Comentarios Finales



1. Metodología de la Programación

Computadora

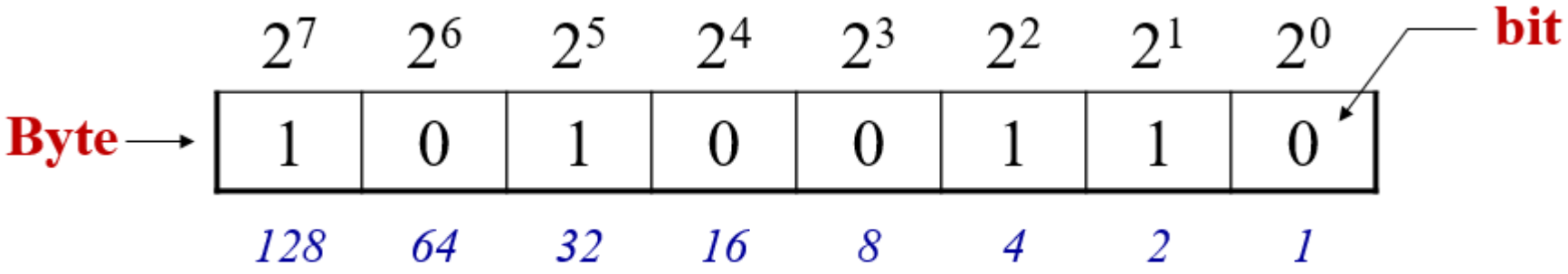
Es una máquina capaz de procesar datos o información; utilizando un programa almacenado

Programa: Conjunto de instrucciones que la computadora ejecuta paso a paso.

Memoria RAM (Memoria de Acceso Aleatorio). Dispositivo electrónico que permite almacenar la información. La unidad básica de almacenamiento es el Byte.

1.1. Almacenamiento de la Información

El almacenamiento de la información en la memoria RAM y en el disco es **Binario**; la unidad básica de almacenamiento se denomina bit(0,1) la combinación de 8 bits forman un Byte.



Estado del Byte = 128+0+32+0+0+4+2+0 = **166**

No.Máx = 2^8-1 = 256-1 = **255**

$$No.Máx = 2^{NoBits} - 1$$

Tipos de Datos

Los datos enteros son los más fácil de representar, se encuentran enteros de 1,2,4 y 8 bytes

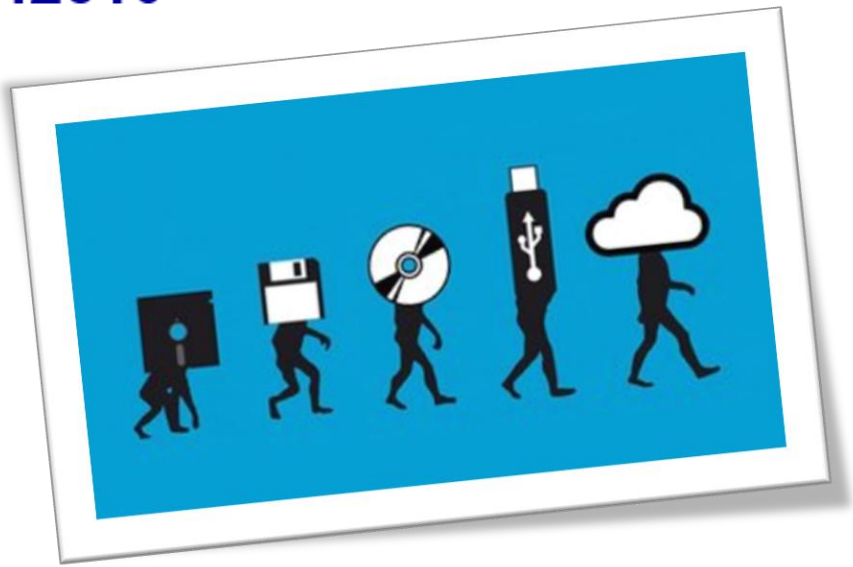
$$No.Máx = 2^{NoBits} - 1$$

2^{15}	2^{14}	2^{13}	2^{12}	2^{11}	2^{10}	2^{9}	2^{8}	2^{7}	2^{6}	2^{5}	2^{4}	2^{3}	2^{2}	2^{1}	2^{0}
1	0	1	0	0	1	1	0	0	0	0	0	1	1	1	0

Valor: 32768+0+8192+0+0+1024+512+0+0+0+0+0+0+8+4+2+0 = **42510**

$No.Máx^{(16)} = 2^{16} - 1 = 65536 - 1 =$ **65535**

$No.Máx^{(32)} = 2^{32} - 1 = 4,294'967,296 - 1 =$ **4,294'967,295**



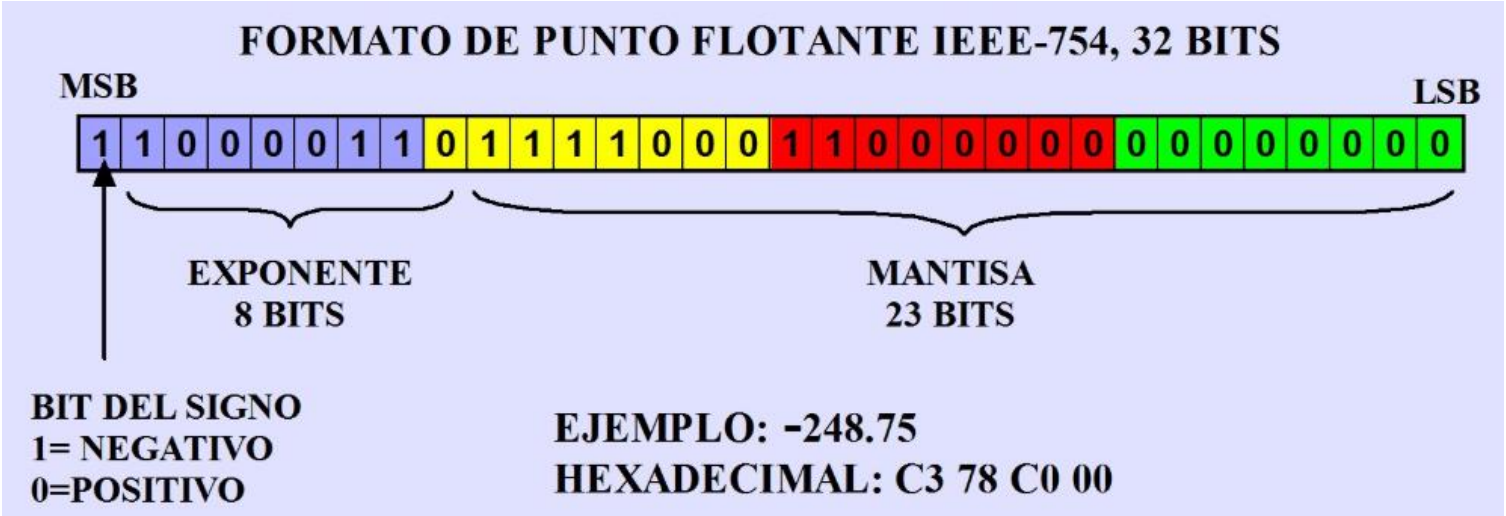
Tipos de Datos

En general existen dos tipos de datos: Numéricos y Alfanuméricos (Cadenas o Texto).

- Numéricos: ***Enteros*** y Valores en ***Punto Flotante***.
- Texto: Caracteres y Cadenas de Texto.

Punto Flotante (Reales): Exponente + Mantiza

Ejemplo para 4 bytes:



Cadenas y Caracteres:

Las cadenas de caracteres forman con símbolos del Código ASCII ó Código EBCDIC.

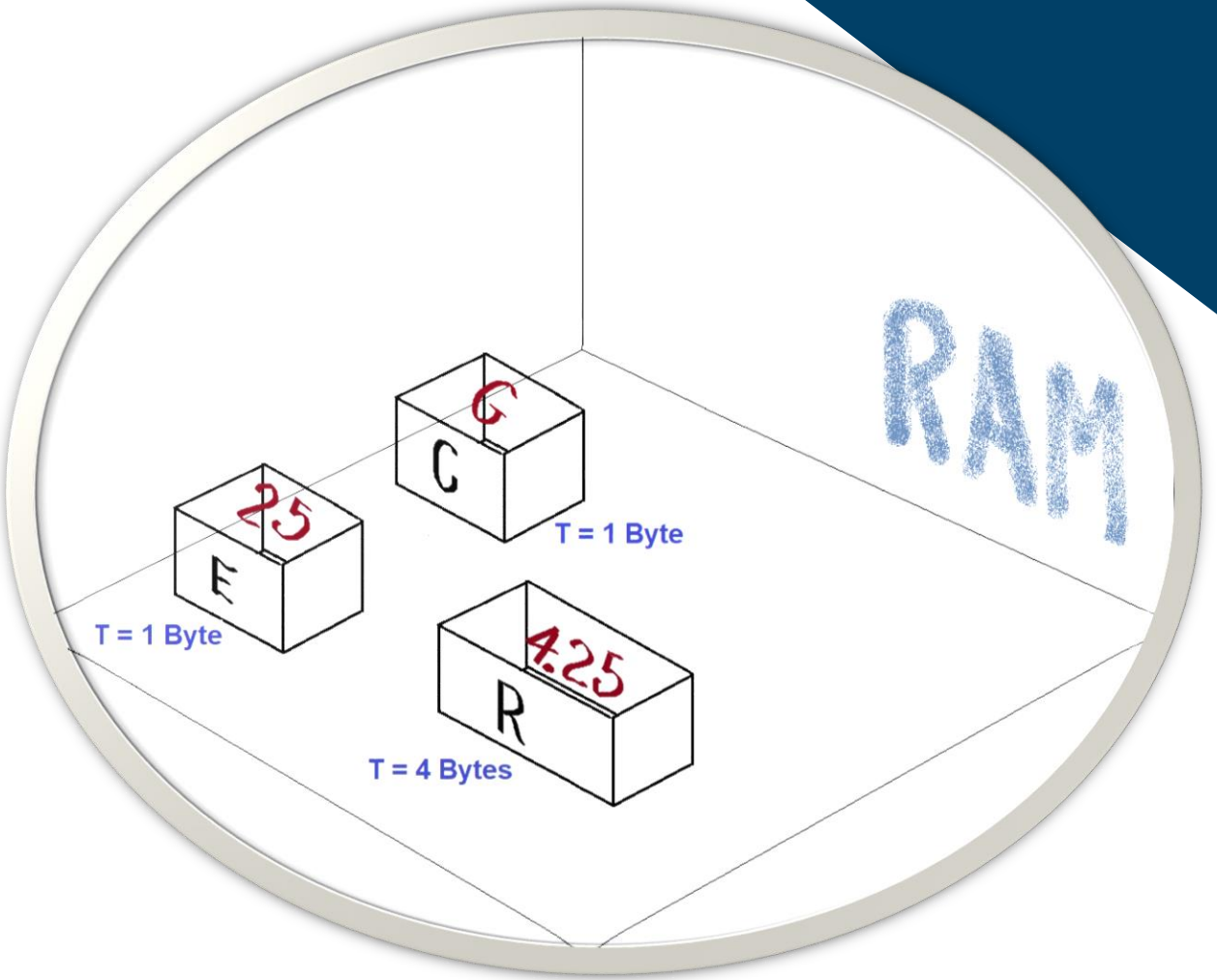
1 Byte por Carácter

Las Variables

Una variable es un espacio reservado en la Memoria que permite almacenar información.

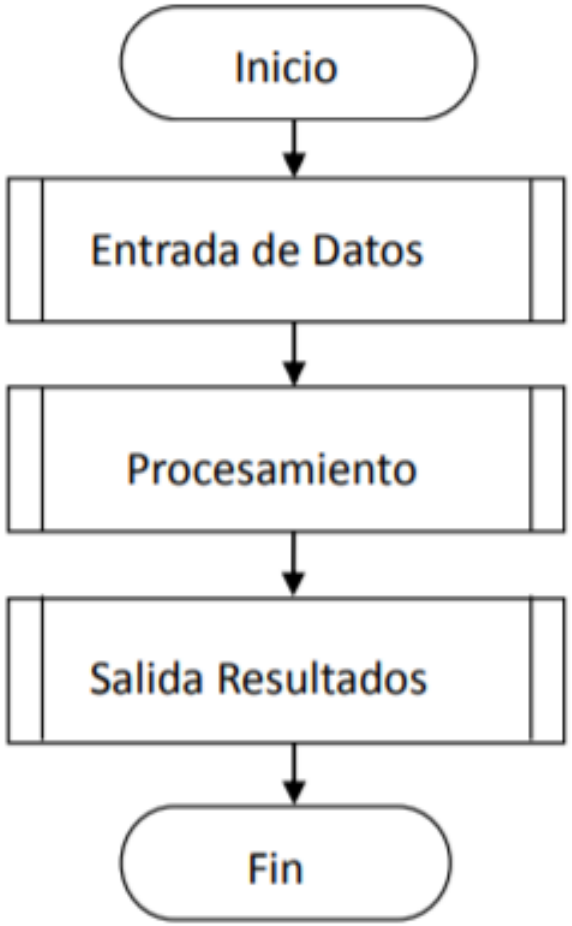
Cualquier Variable debe tener las tres características siguientes:

- Identificador (Nombre)
- Tipo de información (Contenido)
- Espacio que ocupa en la memoria (Tamaño)



Fases de un programa

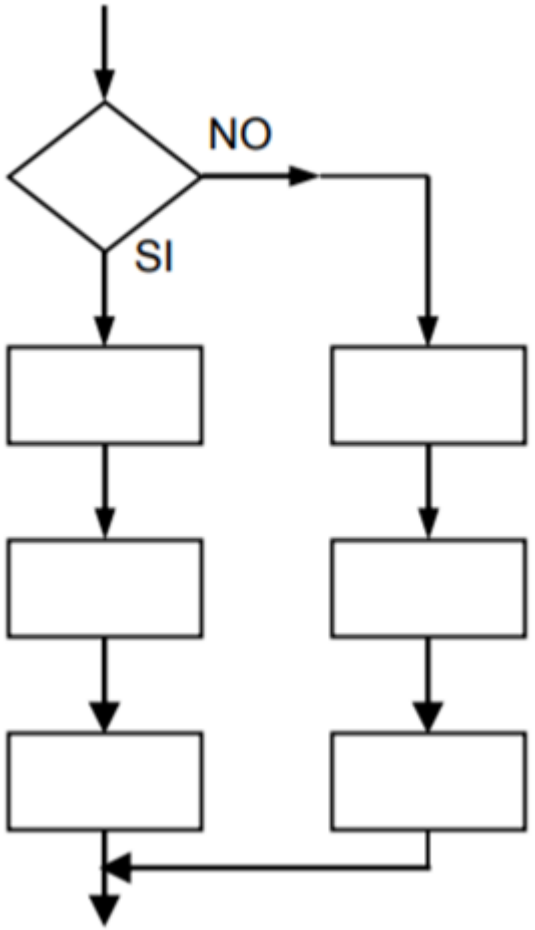
En general todo programa de computadora debe cumplir con las siguientes fases:

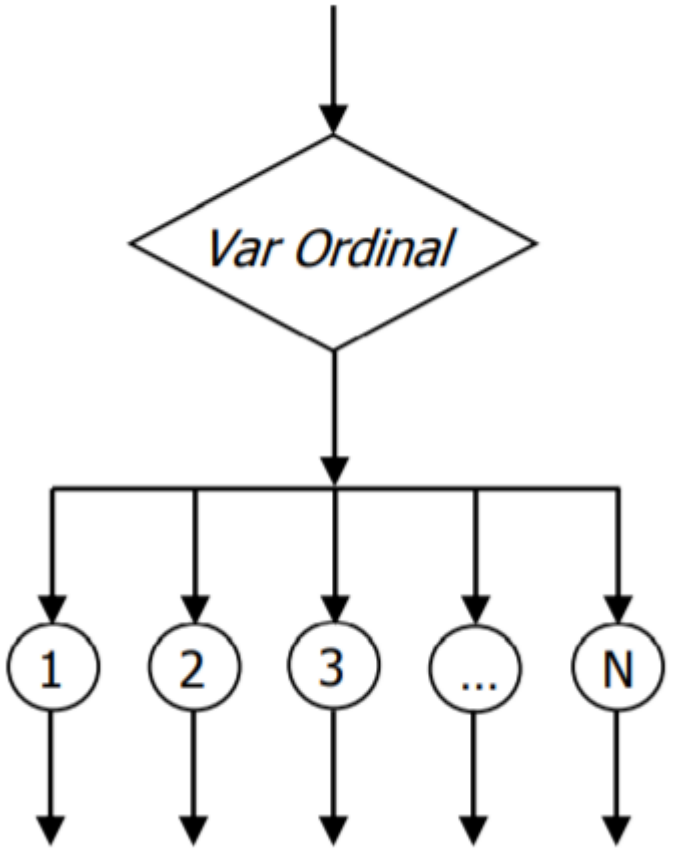


1. *Inicio del programa.*
2. *Entrada de datos.*
3. *Procesamiento.*
4. *Salida de datos.*
5. *Fin.*

Las Estructuras de Control

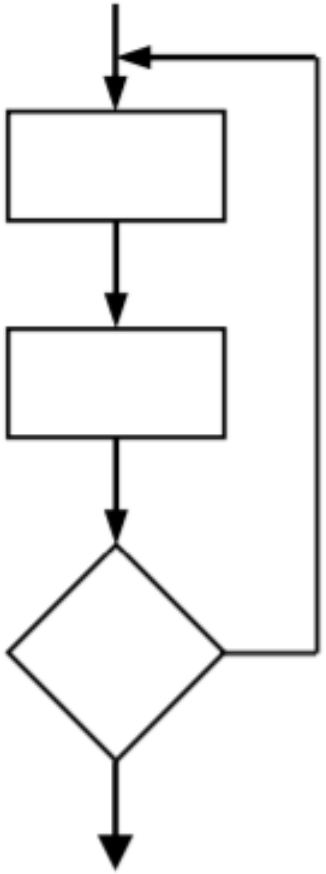
- Condicionales o Selectivas:

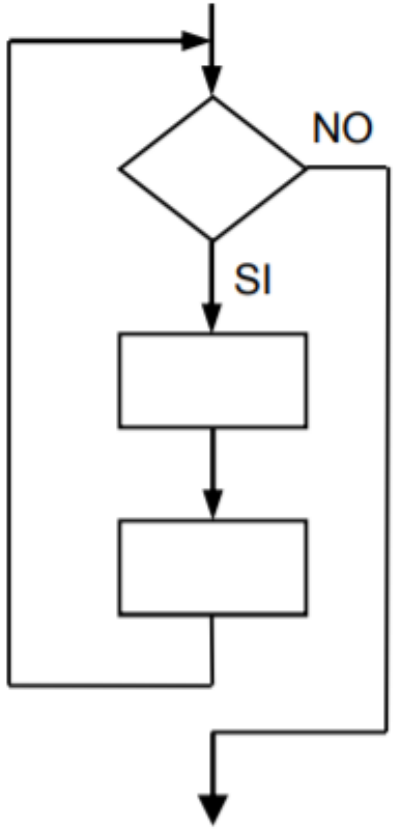


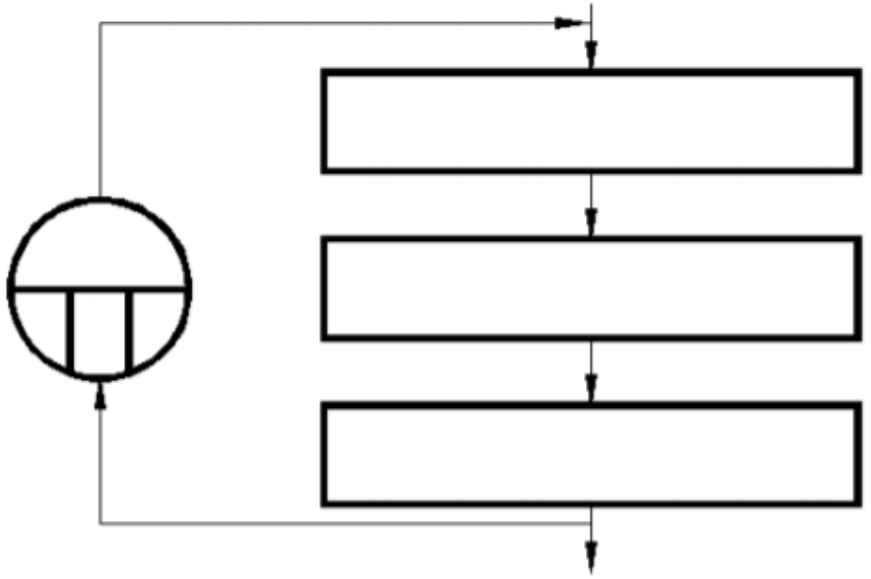


Las Estructuras de Control

- Repetitivas:







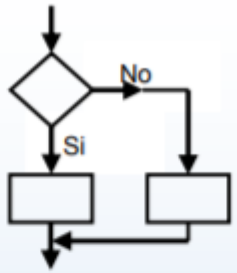
Las Estructuras de Control

Las estructuras de control permiten cambiar el sentido de flujo del algoritmo.

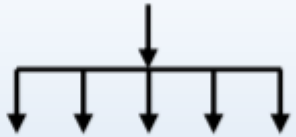
Resumen de las Estructuras de Control:

a. Condicionales

If

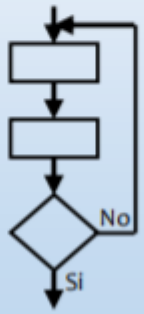


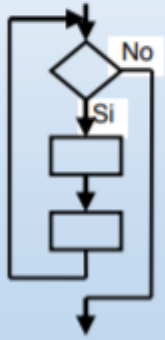
Case
Select Case
Switch

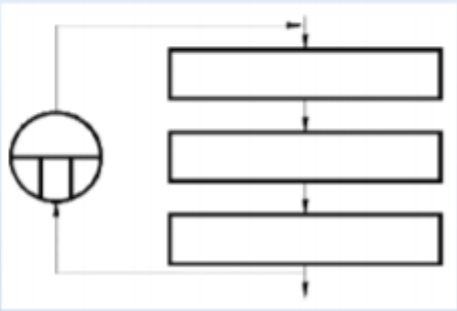


b. Repetitivas

Repeat until/Do while
while
For







Programación Módular

Implica dividir el problema en sub-problemas, que permitan resolverlos de una forma más sencilla. La suma de las pequeñas soluciones generará la solución completa del problema.

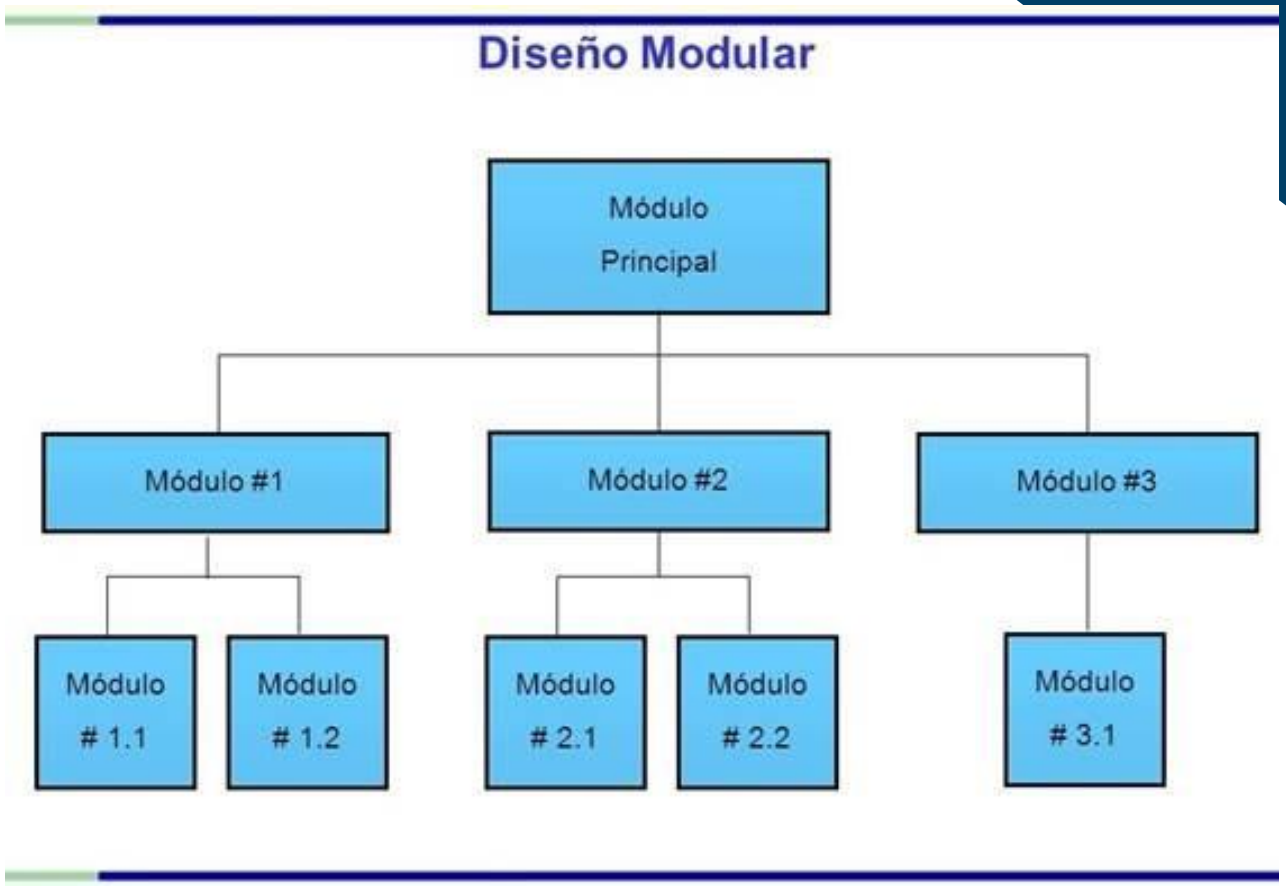
Programa {
Subprograma_1;
Subprograma_2;
.
.
.
Subprograma_N;

Es el corazón de la programación; significa:

"Tomar un problema y hacerlo pedacitos".

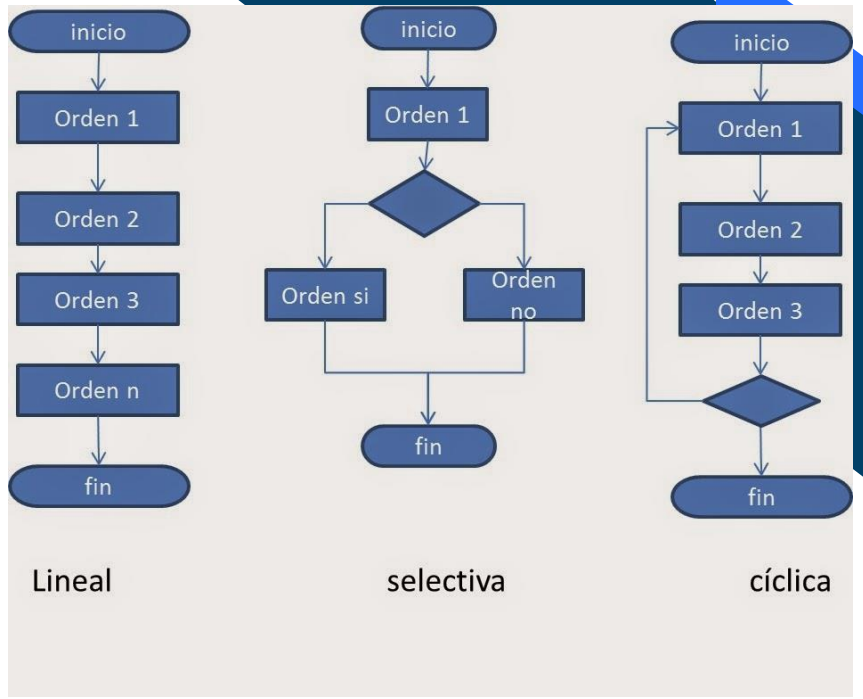
Está basado el concepto en la filosofía ordinaria:

"Divide y Vencerás"

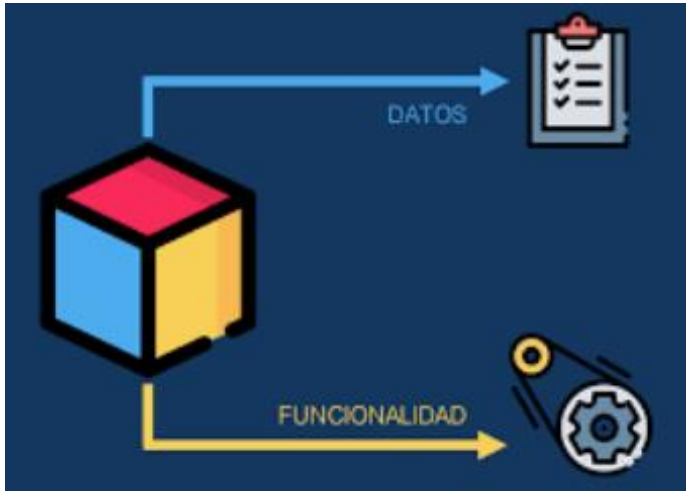


El paradigma de la programación

Se refiere a cual será la filosofía de programación a usar.



- Programación de saltos.
- Programación Estructurada y Modular.
- Programación Orientada a Objetos.
- Programación de Bloques.



La forma en que se **organizan** las **Variables en memoria** definen el **estilo de programación.**

1.2. Pasos básicos para programar

En la Metodología de la programación se tienen definidos los siguientes pasos para programar:

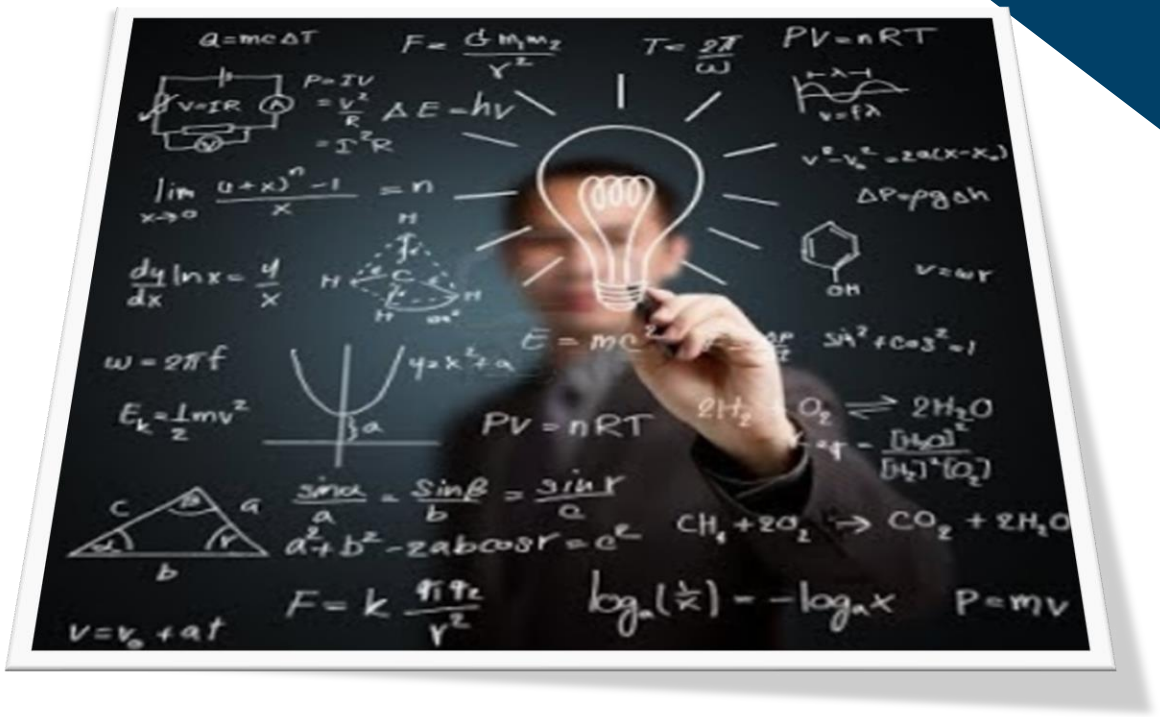
- Análisis del programa
- Diseño del algoritmo
- Prueba del algoritmo
- Codificación del algoritmo
- Ejecución, Prueba y depuración del código.
- Documentación del programa
- Mantenimiento del programa

Algoritmo: Conjunto de pasos ordenados y sistematizados que conducen a la solución de un problema

Análisis del problema

Parte medular de la programación:

- ¿Tengo el Conocimiento suficiente para resolver el problema?
- Variables de Entrada
- Variables de proceso
- Variables de Salida
- Variables Auxiliares



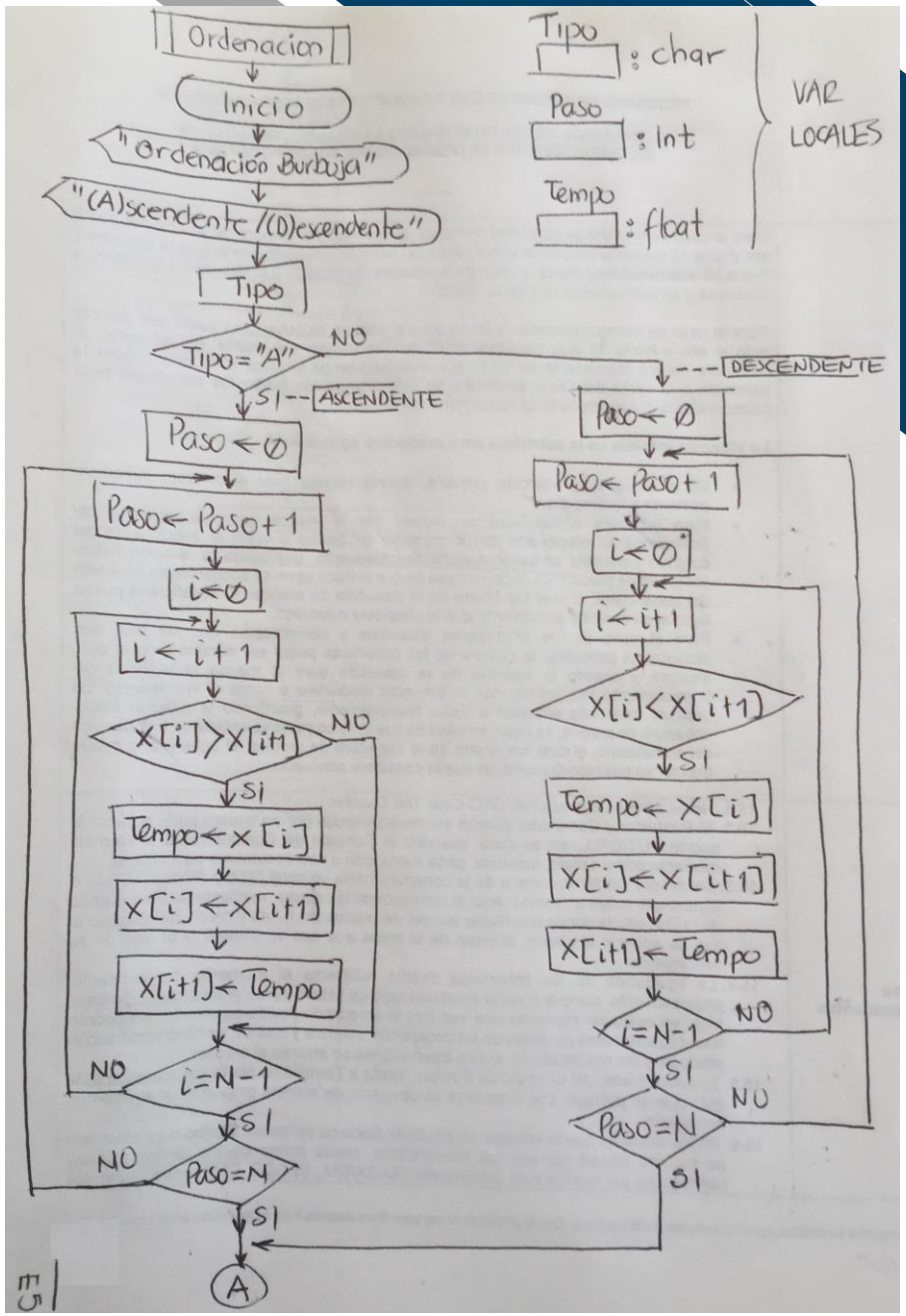
Diseño del Algoritmo

Conjunto de pasos ordenados y sistematizados que conducen a la solución de un problema.

- Preciso
- Definido
- Finito

Formas de Representación:

- Pseudocódigo
- Diagrama de Flujo



1.3. El código fuente

Depende del lenguaje de programación a utilizar, y está constituido por las siguientes reglas:

- De Sintaxis
- De Semántica
- Símbolos y Palabras Reservadas

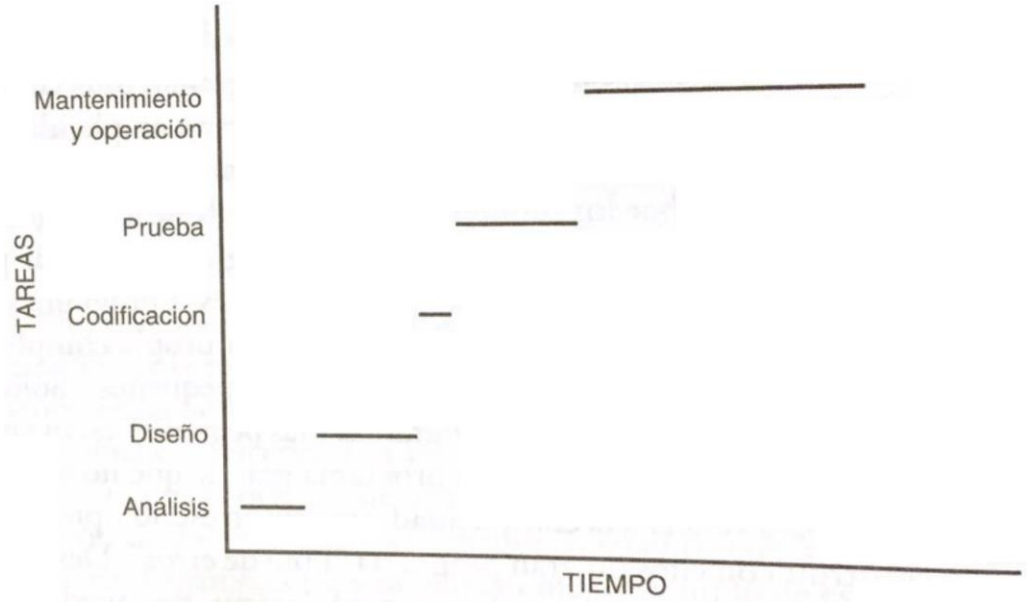
El Código Fuente: Se escribe por lo general en un editor de código ASCII. Para el programador representa la codificación del algoritmo, pero para la computadora … NADA.

```
void Ordena(char Tipo)
{
   int   i,Paso;
   float Tempo;

   if (Tipo=='A') {
     // Ordenacion Ascendente
     Paso=0;
     do {
        Paso++;
        i=0;
        do {
           i++;
           if (x[i]>x[i+1]) {
              Tempo=x[i];
              x[i]=x[i+1];
              x[i+1]=Tempo;
           }
        } while (i<(N-1));
     } while (Paso<N);
   }
   else {
     // Ordenacion Descendente
     Paso=0;
     do {
        Paso++;
        i=0;
        do {
           i++;
           if (x[i]<x[i+1]) {
              Tempo=x[i];
              x[i]=x[i+1];
              x[i+1]=Tempo;
           }
        } while (i<(N-1));
     } while (Paso<N);
   }
}
```

1.4. Puesta a punto de un programa

El tiempo en el que se desarrolla un algoritmo representa entre el 20-30% del tiempo de desarrollo de la aplicación. El restante (70-80%) se invierte en hacer que el programa sea funcional y práctico para el usuario.

- El mantenimiento de la aplicación deberá ser continua y permanente.

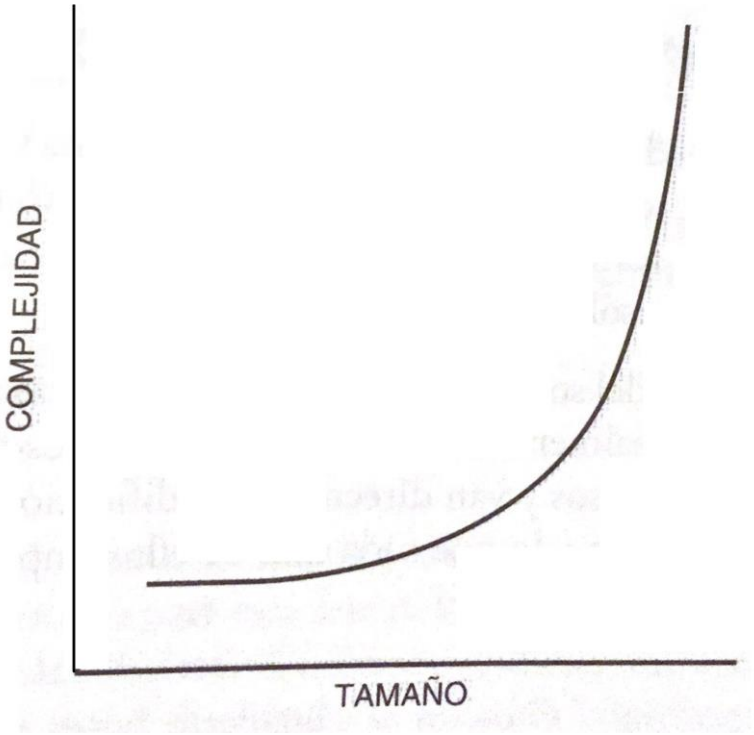


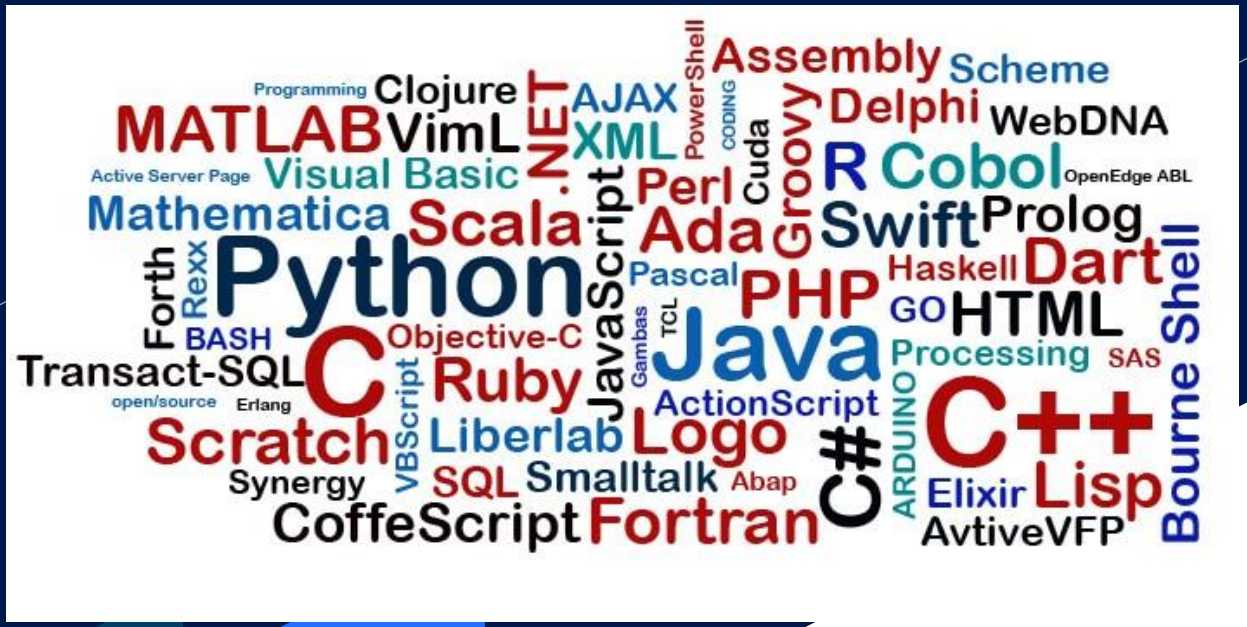
El ciclo de vida del software.

1.5. Los errores clásicos al programar

El mayor error encontrado en los programadores, es que se saltan los pasos de la metodología de la programación. Lo que afecta grandemente el desarrollo de cualquier programa.

Tiempo de Programación: A mayor complejidad del programa, aumenta el tiempo de desarrollo.





2. Lenguajes de Programación

2.1. Definición y tipos de lenguajes

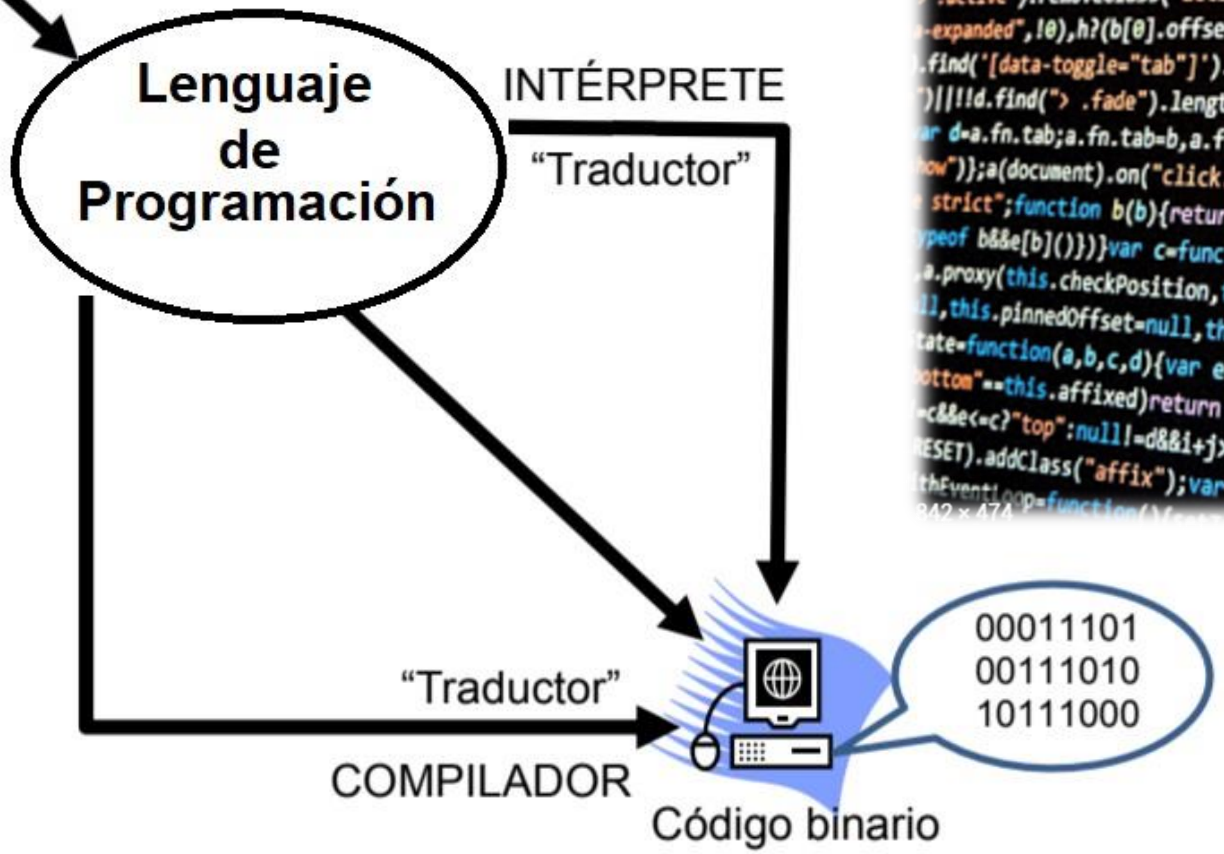
Un **lenguaje de programación** es un conjunto de reglas de sintaxis, de semántica, de símbolos y palabras reservadas que permiten la comunicación entre el usuario y la computadora.

- Basic → Visual Basic
- Pascal → Delphi
- C → C#, Builder

Derivados de C: JavaScript, Phyton, PHP, Arduino, Mathlab,etc.

Lenguajes de Programación

Lenguaje común



Lenguajes de Programación

Lenguaje común

Declarativos

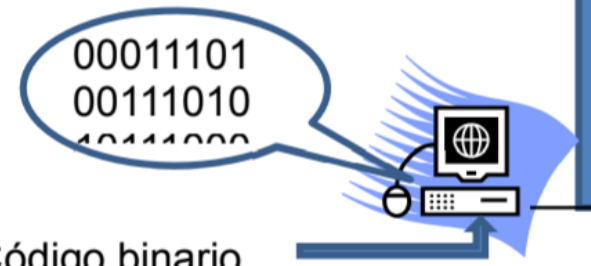
Lenguaje de uso común
Son transportables
***Usan** herramientas prefabricadas*

Alto nivel

Lenguaje de uso común
Son transportables
*Se **hacen** las herramientas*

Ensambladores

Usas símbolos "Nemotécnicos"



Código binario

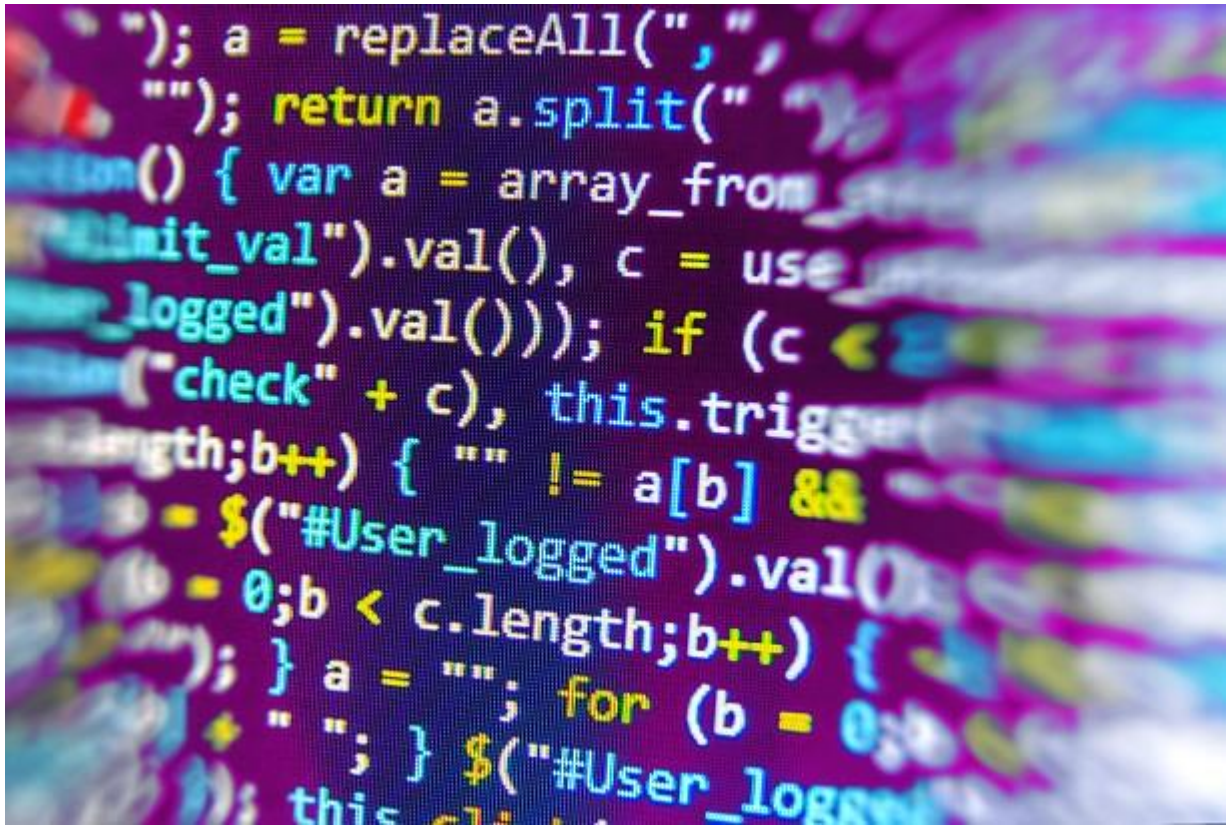
Bajo nivel o máquina

Usa código binario
No son transportables

Los más populares

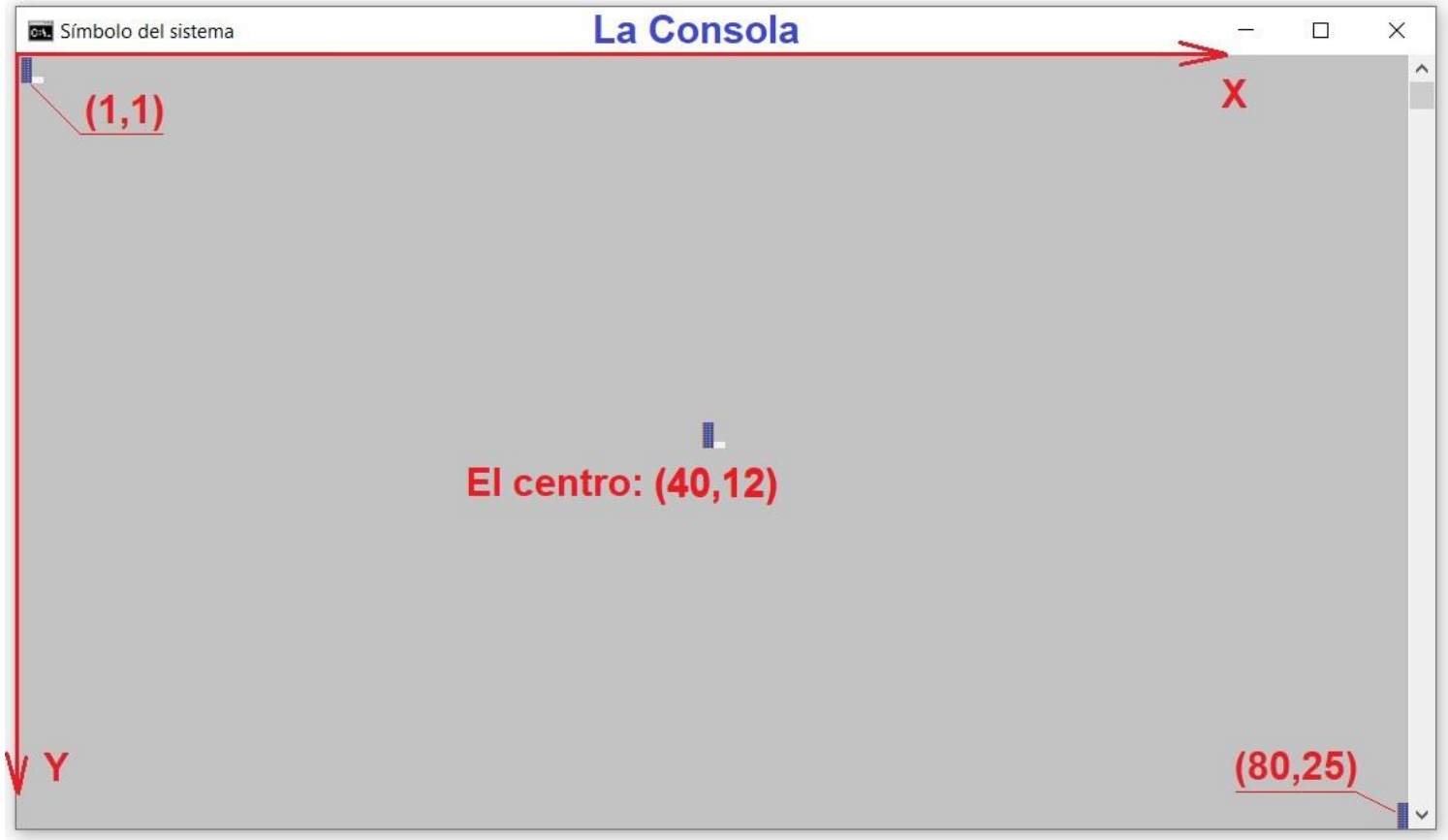
May 2019	Programming Language	Ratings
1	Java	16.005%
2	C	14.243%
3	C++	8.095%
4	Python	7.830%
5	Visual Basic .NET	5.193%
6	C#	3.984%
7	JavaScript	2.690%
8	SQL	2.555%
9	PHP	2.489%
10	Assembly language	1.816%

El Mejor Lenguaje de Programación



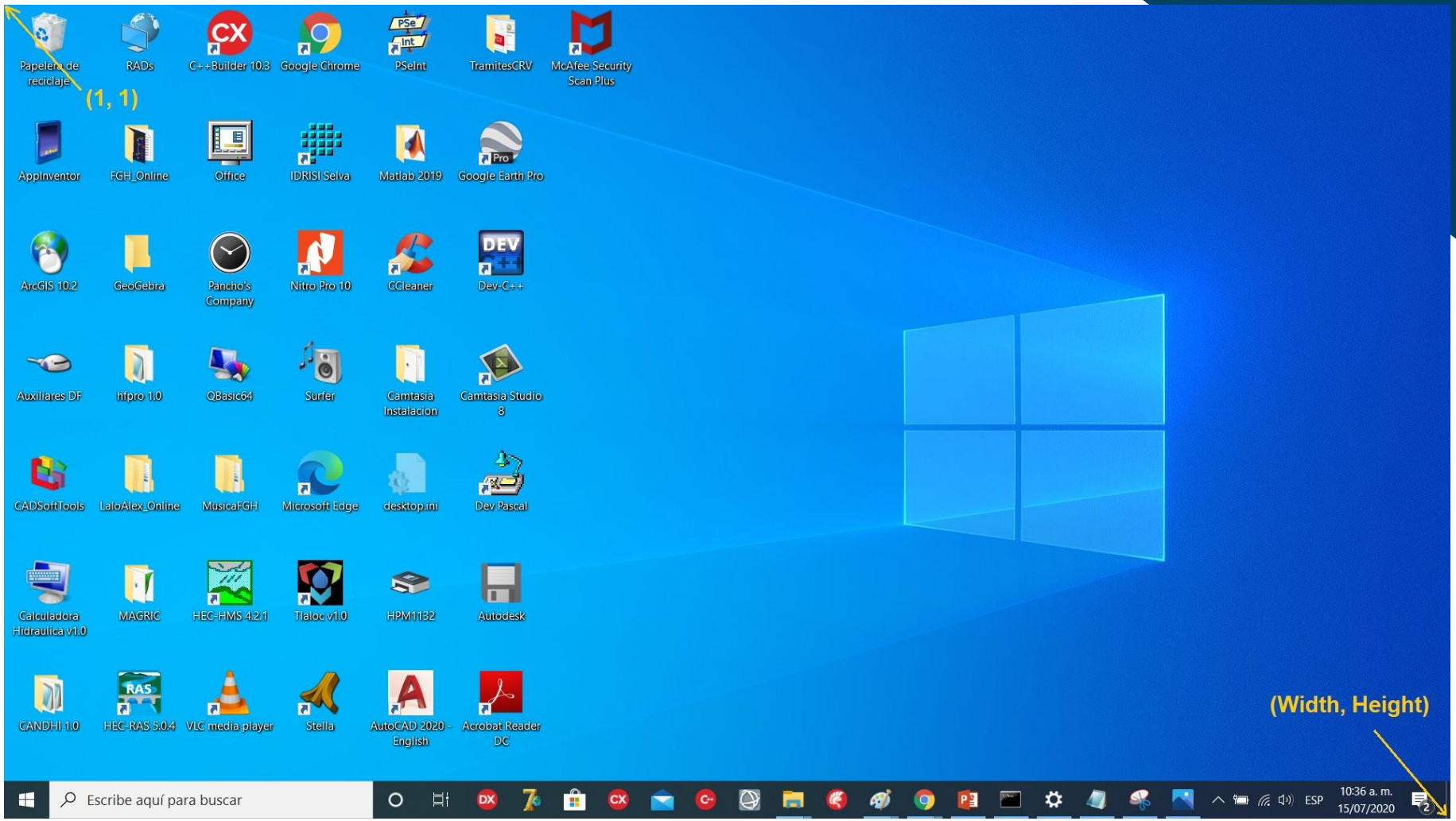
Definitivamente: El mejor lenguaje es el que se domine a la perfección.

2.2. La Consola



Consola: El trabajo en la consola es con Caracteres (Char)

2.3. La GUI (Graphics User Interface)

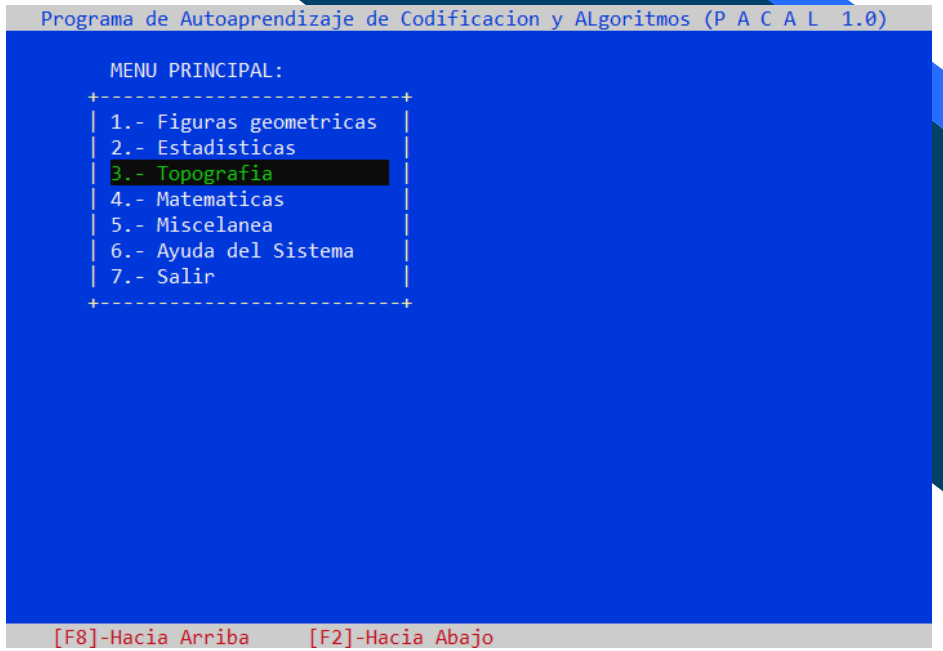


Interfaz Gráfica del Usuario (GUI): El tamaño de la GUI depende de la resolución de la tarjeta gráfica.

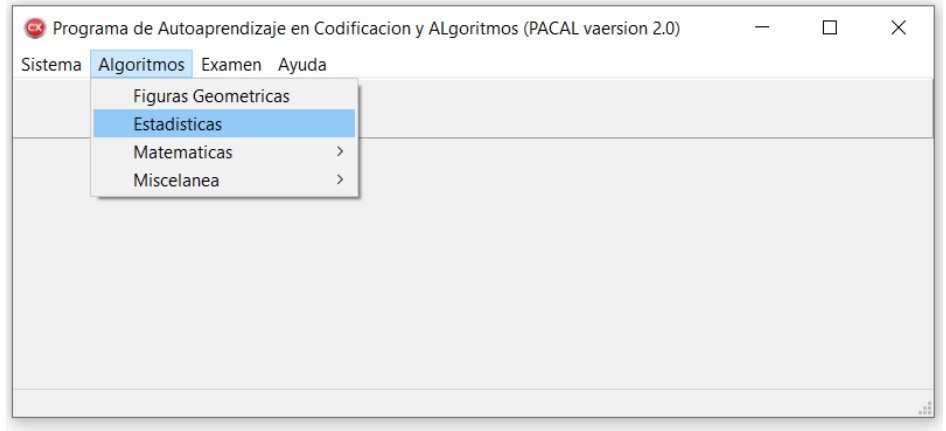
2.4. Consola vs GUI

La diferencia radica en el elemento básico que la compone:

- Consola → Character
- GUI → PIxEL



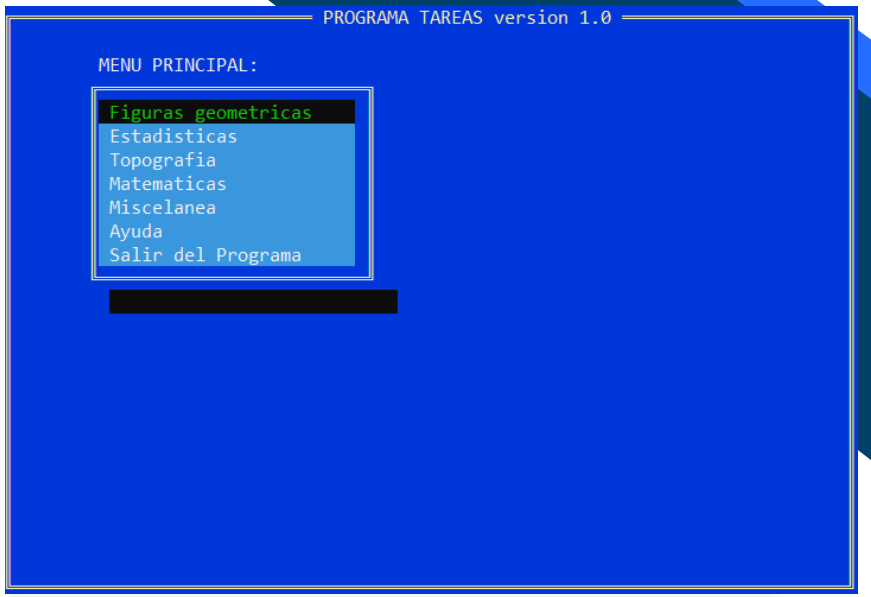
La apariencia: Por su puesto que la GUI permite hacer programas con elementos gráficos que la CONSOLA no puede tener. Pero el algoritmo de cálculo será el mismo.

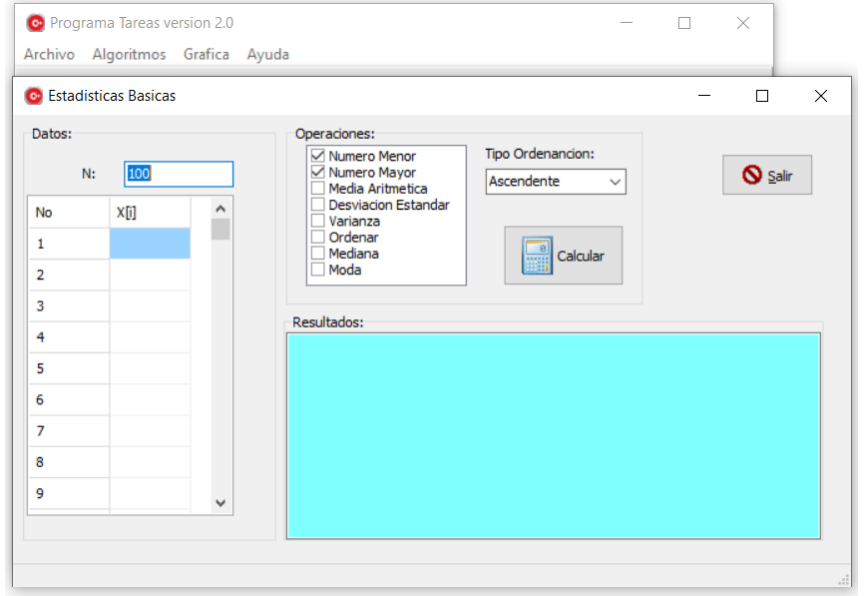


Consola vs GUI

La entrada y salida de información será distinta.

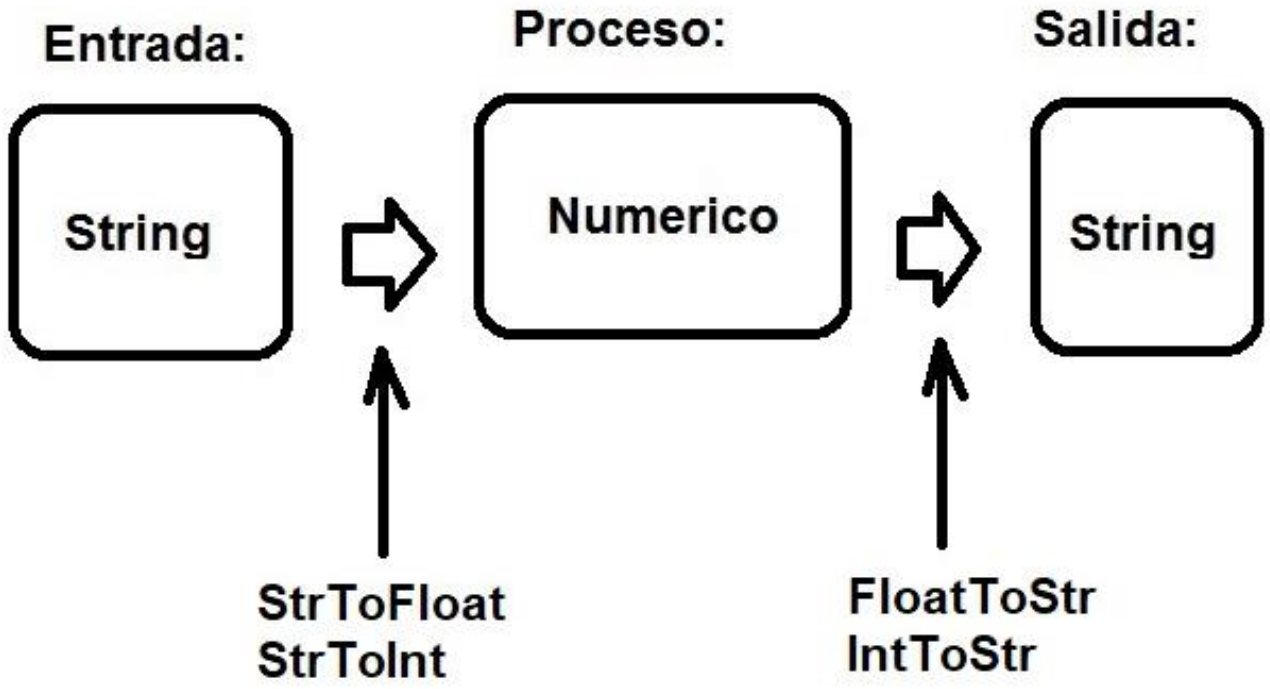
- Consola → Numérica y Alfanumérica
- GUI → Alfanumérica





En ambos casos el **procesamiento** de la información en **Numérico**.

Consola vs GUI

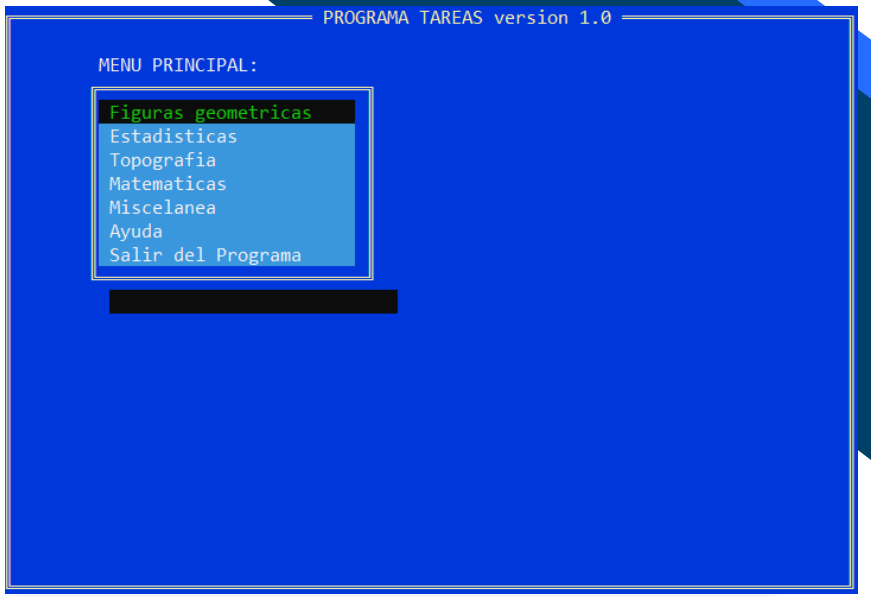


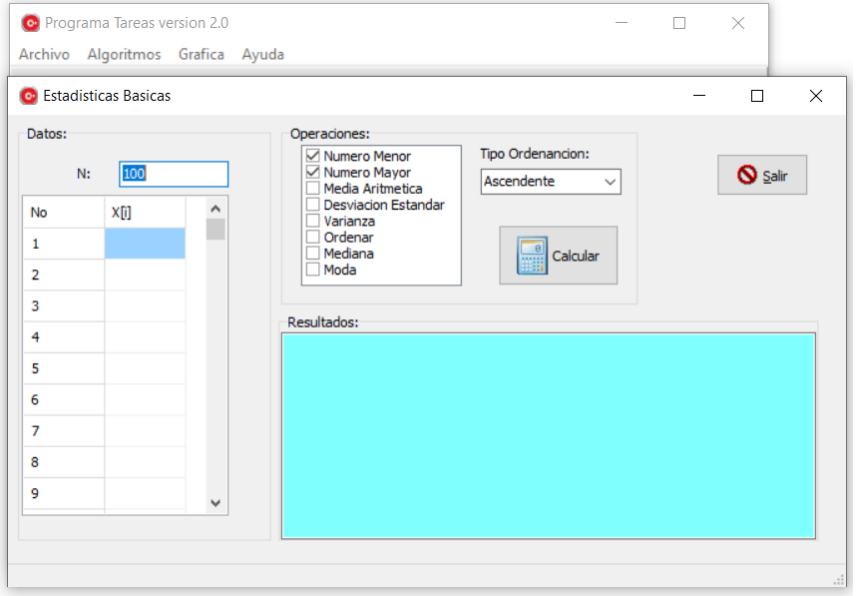
StrToFloat ----> Cadena a Real Flotante

StrToInt ----> Cadena a Entero

IntToStr ----> Entero a Cadena

FloatToStr ----> Real Flotante a Cadena





3. Desarrollo de Aplicaciones

3.1. Elección de la plataforma de desarrollo

La elección de la plataforma de desarrollo depende del tipo de aplicación a realizar.

Privada: Cualquiera es funcional.
Publica: Habrá que considerar el destino final, si estará en RED, móvil o será una aplicación de escritorio.

3.2. La facilidad de usar un RAD

RAD: Rapid Application Developed (Desarrollador Rápido de Aplicaciones)

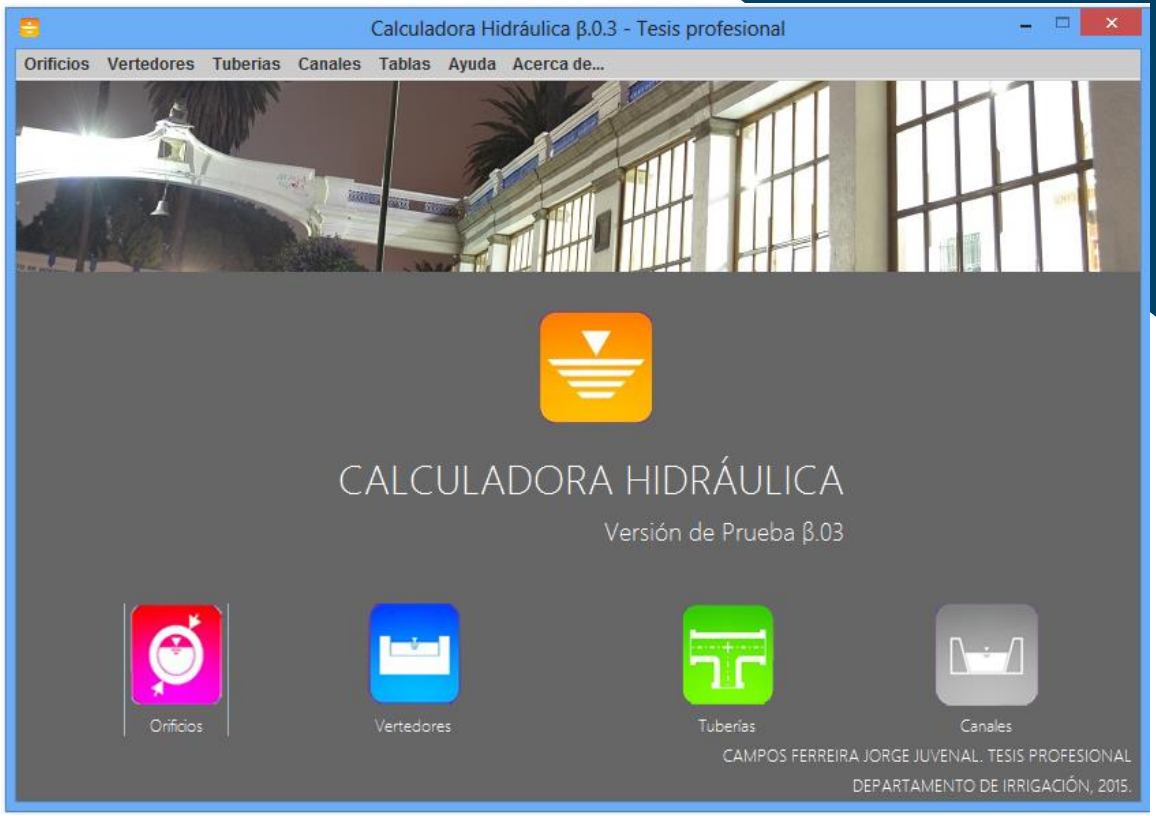
Los RAD modernos se conocen también como Ambientes Visuales, basados en la filosofía de la Programación Orientada a Objetos (POO).

- Basic → Visual Basic
- Pascal → Delphi
- C/C++ → Builder
- C++++ → C#
- Phyton → Anaconda

3.3. Tipo de aplicaciones en la ingeniería

En la ingeniería las aplicaciones que se deben desarrollar, son especializadas en resolver problemas específicos:

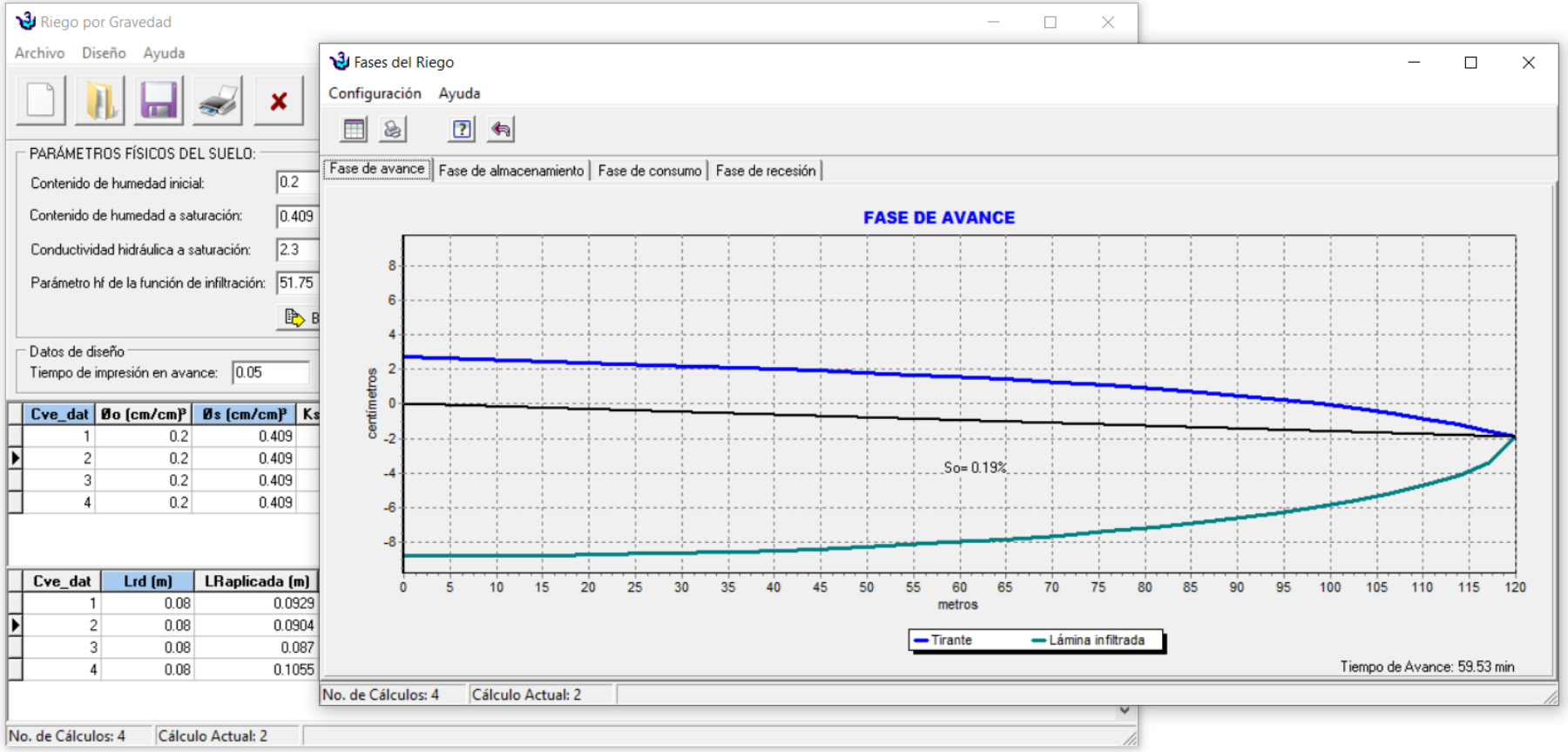
- Matemáticos
- Meteorológicos
- Estadísticos
- Hidrológicos
- Hidráulicos
- Termodinámicos
- De Mecánica
- Entre otros.



Aplicaciones: Es poco probable encontrar una aplicación que satisfaga el problema en forma total.

3.4. Utilidad de las aplicaciones a la medida

Cuando se domina la programación y un lenguaje para codificar, las aplicaciones son trajes a la medida, que resuelven el problema a satisfacción.



Recomendación: Es necesario conocer y dominar algún lenguaje de programación para facilitar el trabajo de análisis, planeación y diseño.

3.3. Tipo de aplicaciones en la ingeniería

En la ingeniería las aplicaciones que se deben desarrollar, son especializadas en resolver problemas específicos:

- Matemáticos
- Meteorológicos
- Estadísticos
- Hidrológicos
- Hidráulicos
- Termodinámicos
- De Mecánica
- Entre otros.


```
Public Function ff(Q As Single, D As Single, L As
Single, e As Single) As Double
Dim Tolerancia As Single
Dim NoIterMax As Byte
Dim A As Single, v As Single
Dim Rr As Single, Re As Single
Dim Vcine As Single
Dim fo As Double, f1 As Double
Dim Salir As Boolean
Dim i As Integer

Const pi = 3.14159265358979
Vcine = 0.000001011
Tolerancia = 0.0000001
NoIterMax = 100
```

```
A = (pi * (D) ^ 2) / 4
v = Q / A
Re = (v * D) / Vcine
Rr = e / (D * 1000)
fo = 64 / Re    'Propuesta inicial: Reynolds
Salir = False
i = 0
Do
  f1 = f_G(Rr, Re, fo)
  If (Abs(f1 - fo) <= Tolerancia) Then
    MsgBox "Se llego a la solucion en " _
          & " iteraciones." & Chr(13) _
          & "f = " & f1)
    ff = f1
    Salir = True
  Else
    i = i + 1
    fo = f1
  End If
Loop Until ((Salir = True) Or (i >= NoIterMax))
If (i >= 100) Then
  MsgBox ("El Metodo FALLO. Cambie de Metodo")
  ff = -999.999
End If
End Function

Public Function f_G(Rr As Single, Re As Single, fp As
Double) As Double
f_G = 1 / (2 * Log10((Rr / 3.71) + (2.51 / (Re *
Sqr(fp))))) ^ 2
End Function

Public Function Log10(x As Double) As Double
' logaritmo en base 10 (decimal)
    Log10 = Log(x) / Log(10#)
End Function
```

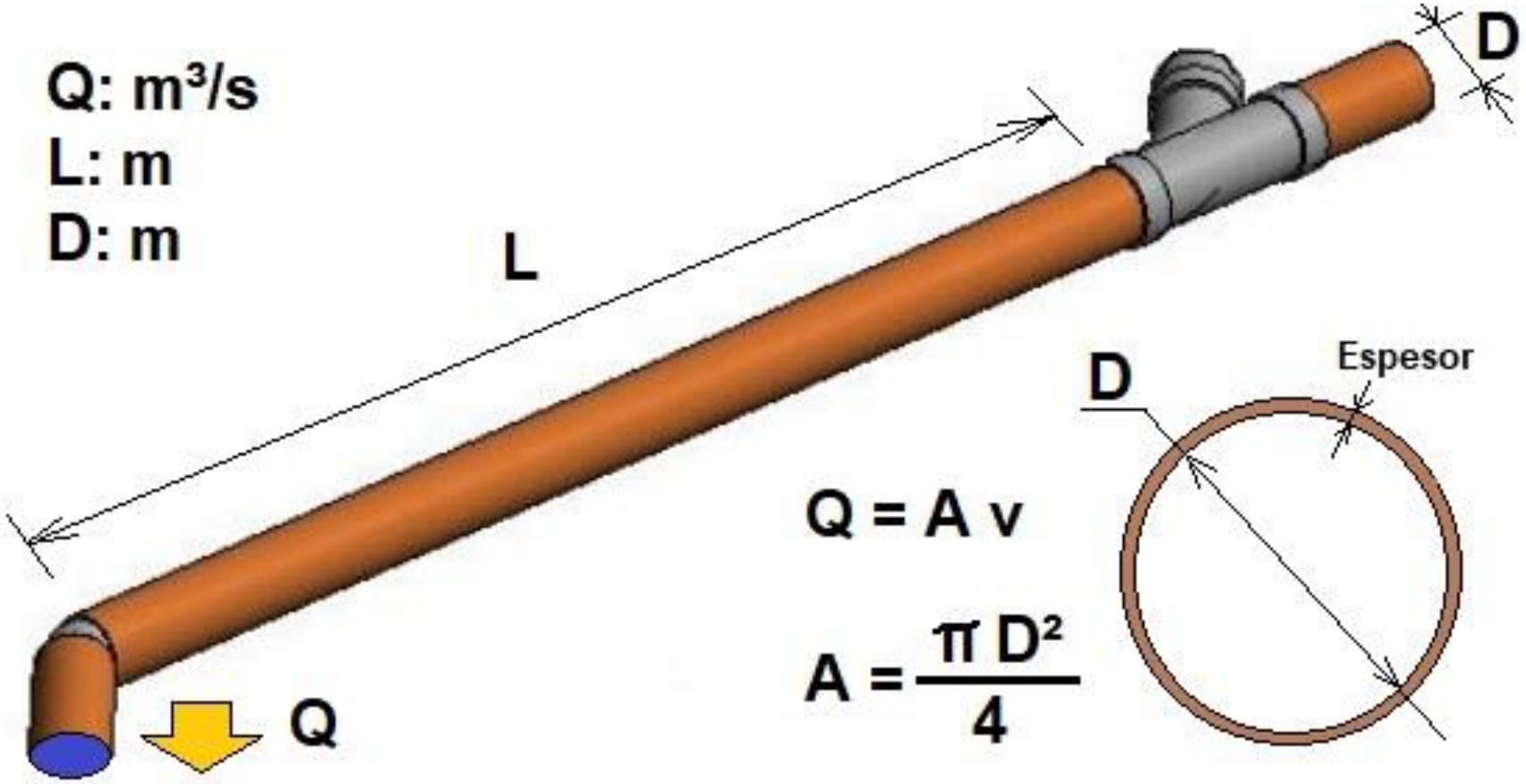
Aplicaciones: Es poco probable encontrar una aplicación que satisfaga el problema en forma total.

Un Ejemplito

Perdidas de Carga en Tuberías

Pérdidas de carga en tuberías

Problema: Diseñar un programa en la Consola y en la GUI de Windows para calcular la pérdida de carga en una tubería con la expresión de Darcy-Weisbach.



Pérdidas de carga en tuberías

- ☑ Darcy Weisbach:

$$hf = f\frac{L}{D}\frac{v^2}{2g}$$

hf = Pérdida de carga en la tubería (m).

f = Factor de fricción (adim).

v = Velocidad del flujo (m/s).

g = Constante de la gravedad (m/s²)

Aunque la siguiente expresión puede ser más practica si se usa el Gasto (Q) que conduce la tubería en lugar de la velocidad (v).

$$hf = 0.082626857\, f\, \frac{Q^2}{D^5}\, L$$

Q = Gasto de la tubería (m³/s).

D = Diámetro de la tubería (m).

L = Longitud de la tubería

Pérdidas de carga en tuberías

El factor de fricción (f), depende del material de la tubería, del número de Reynolds (de la velocidad del flujo, de la viscosidad cinemática del agua, del diámetro), entre otros. Algunos autores que proponen el cálculo de f son:

El número de Reynolds (Re):

$$Re = \frac{v\,D}{\vartheta}$$

Donde:

ϑ= Viscosidad Cinemática del agua (A T20° es 0.00000101)

Coeficientes según material:

No	Material	Rug.(mm)	Chw	n
0	Aluminio con Conexiones	0.0150	130	0.0080
1	Aluminio Conexiones (Tabla Irri	0.0150	140	0.0080
2	Policloruro de Vinilo (PVC)	0.0200	145	0.0090
3	Polietileno (PE)	0.0020	150	0.0080
4	Acero Galvanizado	0.0700	125	0.0105
5	Asbesto Cemento	0.0750	135	0.0110
6	Concreto liso	0.4000	130	0.0125
7	Concreto común	0.9500	120	0.0160
8	Fibrocemento	0.0250	140	0.0115
9	Fierro epóxico	0.0280	145	0.0100
10	Fierro fundido (nuevo)	0.3125	130	0.0145
11	Fierro fundido (15 años)	0.0000	100	0.0000

Rug. = Rugosidad Absoluta (mm)
Chw = Coeficiente de Hazen - Williams
n = Coeficiente de Ruegosidad de Manning

Pérdidas de carga en tuberías

El cálculo de f se hace, en función del régimen del flujo que depende del Número de Reynolds (Re):

- Flujo Laminar (Re<2000):
  - ➔ Poiseville:

$$f = \frac{64}{Re}$$

- Flujo Transicional (2000> Re < 4000):
  - ➔ *Colebrook-White:*

$$\frac{1}{\sqrt{f}} = -2\,log\left(\frac{\varepsilon/D}{3.71} + \frac{2.51}{\sqrt{f}}\right)$$

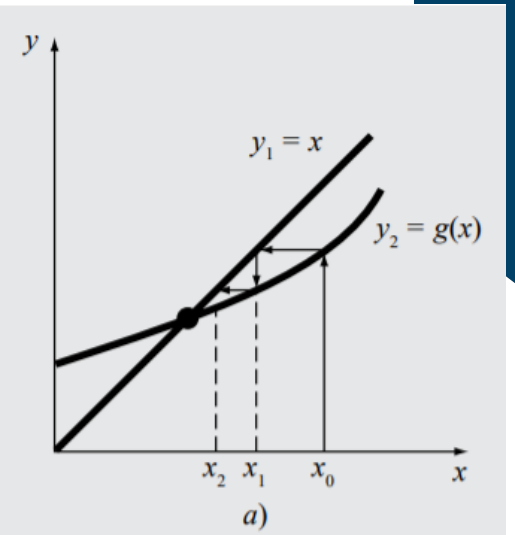
Donde: ε=Rugosidad Absoluta (mm); ε/D=Rugosidad Relativa (adim).

NOTA: Útil también para régimen turbulento.

Pérdidas de carga en tuberías

Para este caso utilizaremos el método del punto fijo y despejaremos parcialmente el valor de f.

$$f = \left[\frac{1}{2 \log\left(\frac{\varepsilon/D}{3.71} + \frac{2.51}{Re\sqrt{f}} \right)} \right]^2$$



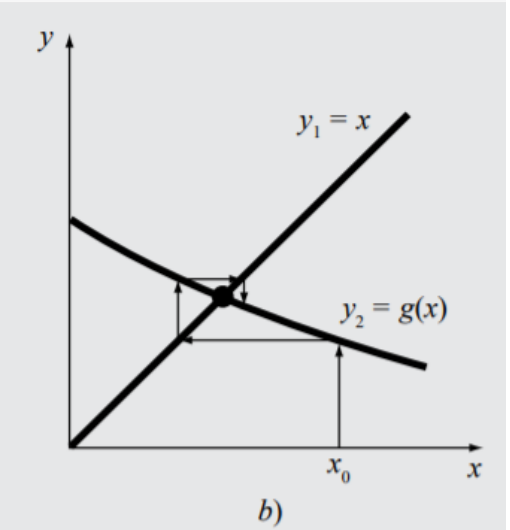
a)

Para lo cual:

x = G(x)

Siendo esta la función G(x) de recurrencia:

$$f_{(i+1)} = \frac{1}{\left[2 \log\left(\frac{\varepsilon/D}{3.71} + \frac{2.51}{Re\sqrt{f_{(i)}}} \right) \right]^2}$$



b)

Algoritmo en pseudocódigo

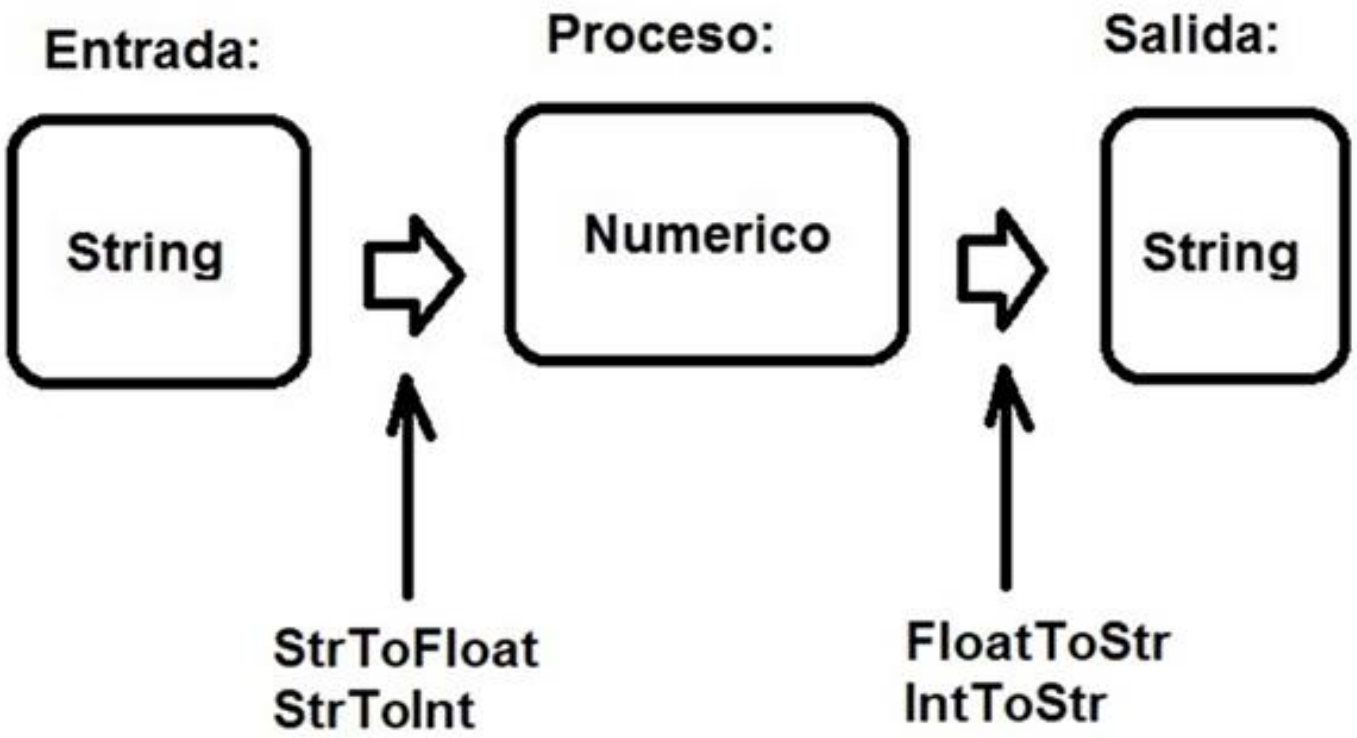
```
CONTANTES:
   Tolerancia = 0.000001;
   NoIterMax = 100;

DATOS DE ENTRADA:
   f(x) --> G(x)
   Q      : Real;   // Gasto (m³/s)
   A      : Real;   // Area (m²)
   e      : Real;   // Rugosidad Absoluta (Tablas)
   D      : Real;   // Diametro (m)
   v      : Real;   // Velocidad (m/s)
   Vcine  : Real;   // Viscosidad cinematica (m²/s)
   Rr     : Real;   // Rugosidad Relativa (e/D)
   Re     : Real;   // Numero de Reynolds
   fo     : Real;   // Propuesta inicial
   f1     : Real;   // Siguiente Aproximacion
```

```
Function f_G(Rr,Re,fp:Real) : Real;
{
    f_G = 1/(2 log10((Rr/3.71)+(2.51/(Re*sqrt(fp)))))^2
}
```

Algoritmo en pseudocódigo

```
CUERPO PRINCIPAL:
A = (pi*D^2)/4;
v = Q/A;
Re = (v*D)/Vcine;
Rr= e/(D*1000)
fo = 64/Re;   // Propuesta inicial f=Poiseville
i=0;
Mientras (i<=NoIterMax) haz
{
  f1=f_G(Rr,Re,fo);
  Si (Abs(f1-fo)<=Tolerancia) Entonces
    {
       Escribe("Llegue a la solucion en ",i,"Iteraciones");
       Escribe("f =",f1);
       SalirCiclo;
    }
  DeOtraForma
    {
      i=i+1;
      fo=f1;
    }
}
Escribe("El método falló, Cambie de método")
```

El proceso a seguir en la GUI



StrToFloat → Cadena a Real Flotante

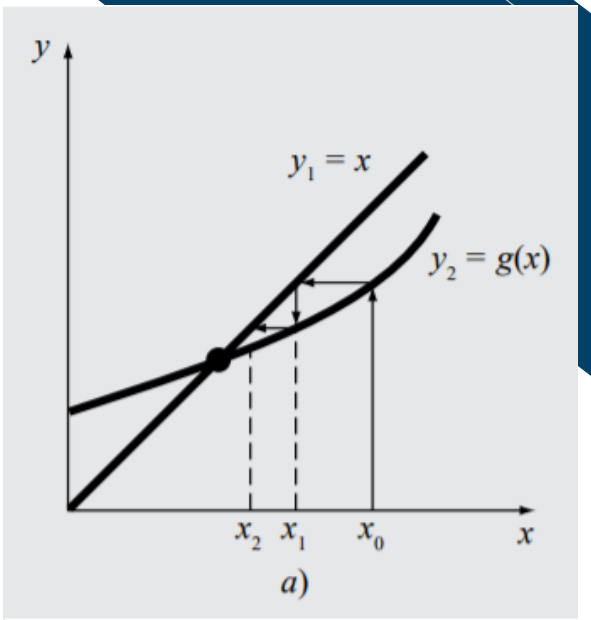
StrToInt → Cadena a Entero

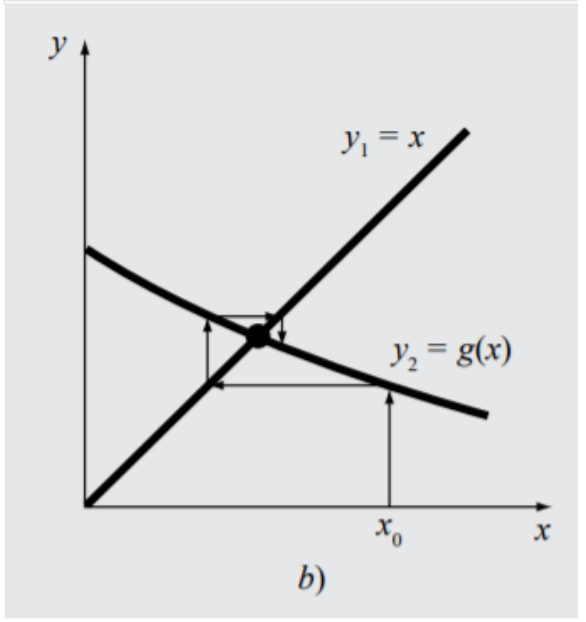
IntToStr → Entero a Cadena

FloatToStr → Real Flotante a Cadena

Pérdidas de carga en tuberías. GUI

```
Function Colebrook_White(Re,Rr:Real):Real;
Const
   Tolerancia = 0.000001;
   NoIterMax = 100;
Var
  f0,f1 : Real;
  i     : Integer;

  Function f_G(Rr,Re,fp:Real) : Real;
  Begin
     // Expresion de Colebrook - White:
     f_G := power(1/(2*log10((Rr/3.71)+(2.51/(Re*sqrt(fp))))),2);
  End;

Begin
   f0:=64/Re;
   i:=0;
   While (i<=NoIterMax) do
   Begin
      f1:=f_G(Rr,Re,f0);
      if (Abs(f1-f0)<=Tolerancia) Then
         Begin
            Colebrook_White:=f1;
            Exit;
         End
      else
         Begin
            i:=i+1;
            f0:=f1;
         End;
   End;
   ShowMessage('El método falló, Cambie de método');
   Colebrook_White:=1;
End;
```



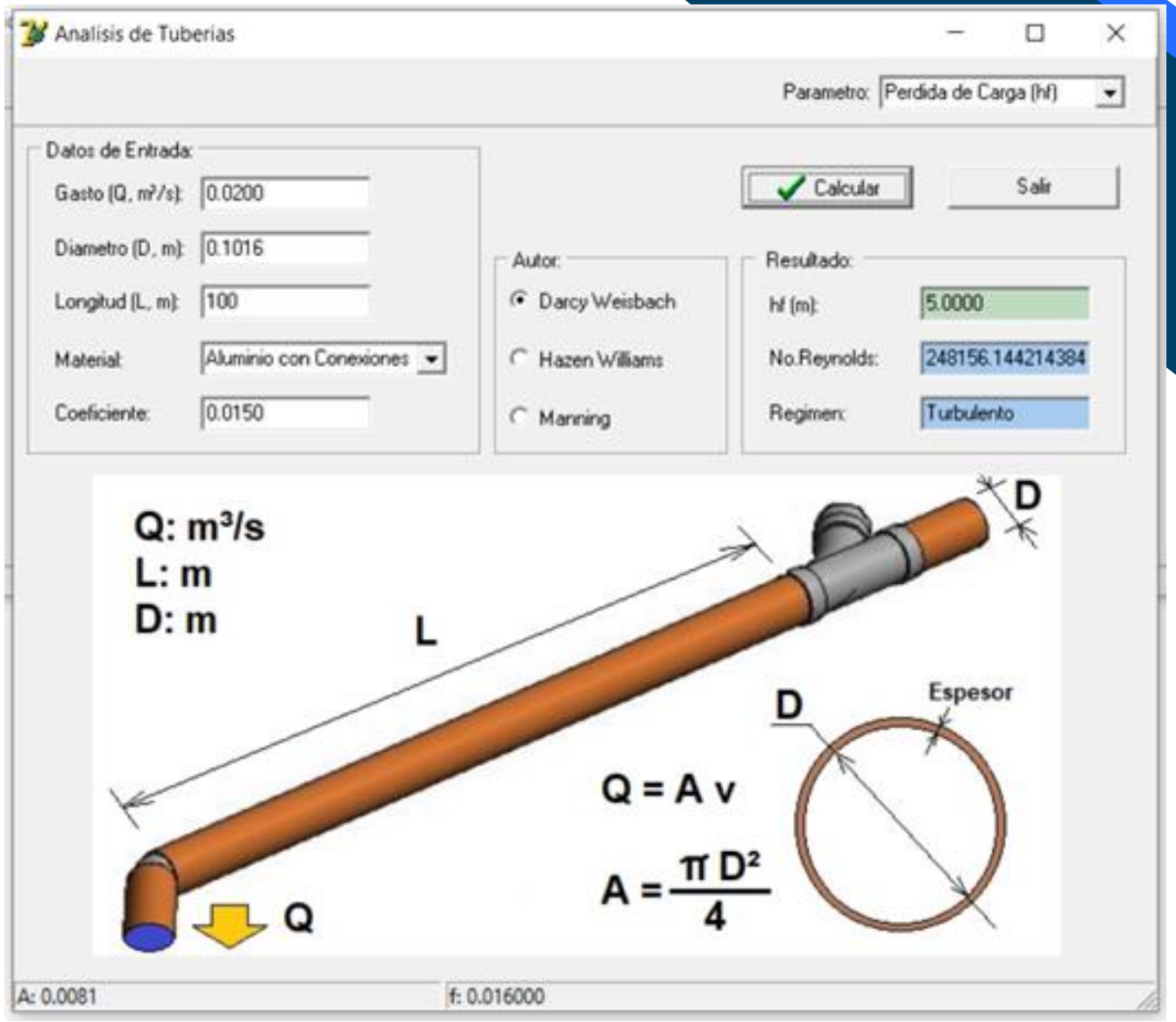


Pérdidas de carga en tuberías. GUI

```
procedure TFormTuberias.BitBtn1Click(Sender: TObject);
Const
   Vcinem = 0.00000101; {a T=20°C}
   Rug : Array[0..11] of Real =(0.0150,0.0150,0.0200,0.0020,0.0700,0.0750,
                                0.4000,0.9500,0.0250,0.0280,0.3125,0.5000);
Var
   Q,D,L,hf,v      : Real;    // Datos Entrada y/o Salida
   f,Re,A,Rr       : Real;    // Variables para el Calculo
   Rug_c           : Real;    // Coeficientes de Rugosidad
   Regimen         : String;

   Procedure Determina_f;
   Begin
      A:=(pi*Sqr(D))/4;
      v:=Q/A;
      Re:=(v*D)/Vcinem;
      Rr:=Rug_c/(D*1000);
      if (Re<=2000) Then   // Flujo Laminar
         Begin
            f:=64/Re; // Poseville
            Regimen:='Laminar';
         End;
      if (Re>2000) And (Re<=4000) Then   // Flujo Transicional
         Begin
            Regimen:='Transicional';
            f:=Colebrook_White(Re,Rr);
         End;
      if (Re>4000) Then  // Flujo Turbuento
         Begin
            Regimen:='Turbulento';
            f:=Colebrook_White(Re,Rr);
         End;
   End;
```

Pérdidas de carga en tuberías. GUI

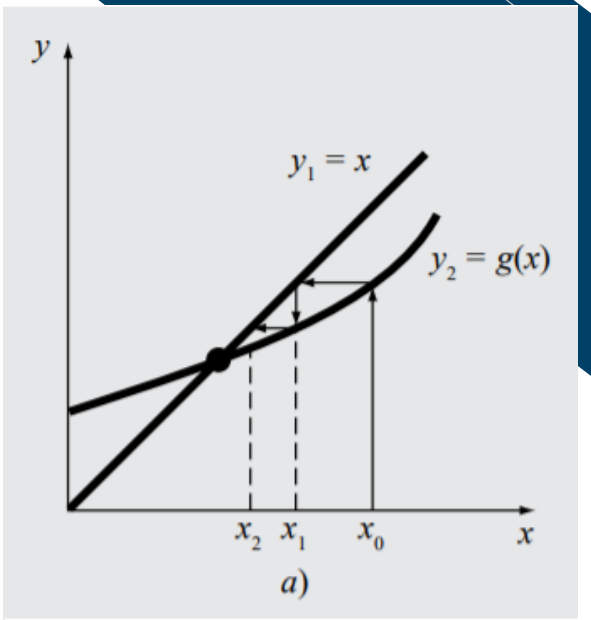
```
// Cuerpo Principal de TFormTuberias.BitBtn1Click
begin
   Rug_c:=Rug[ComboBox1.ItemIndex];
   // Perdidas de Carga (hf)
   Q:=StrToFloat(Edit1.Text);
   D:=StrToFloat(Edit2.Text);
   L:=StrToFloat(Edit3.Text);
   Determina_f;
   hf:=0.082626857*f*(power(Q,2)/power(D,5))*L;
   Edit5.Text:=FloatToStr(hf);
   Edit6.Text:=FloatToStr(Re);
   Edit7.Text:=Regimen;
   StatusBar1.Panels[0].Text:='A: '+FloatToStr(A);
   StatusBar1.Panels[1].Text:='f: '+FloatToStr(f);
end;
```

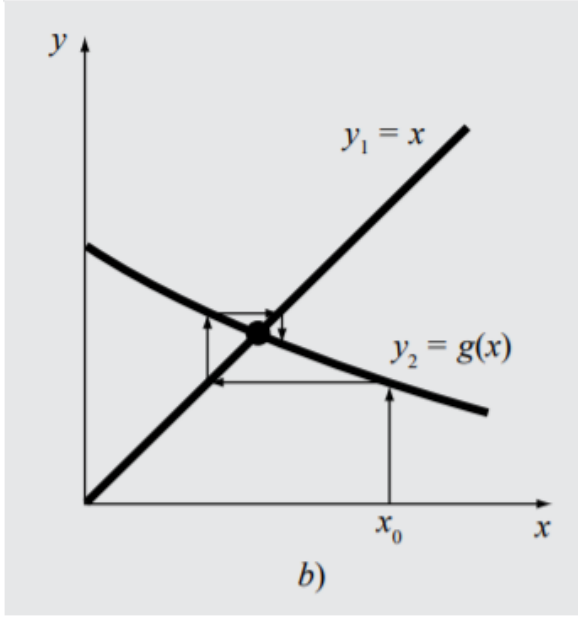


El proceso a seguir en la CONSOLA

Pérdidas de carga en tuberías. Consola

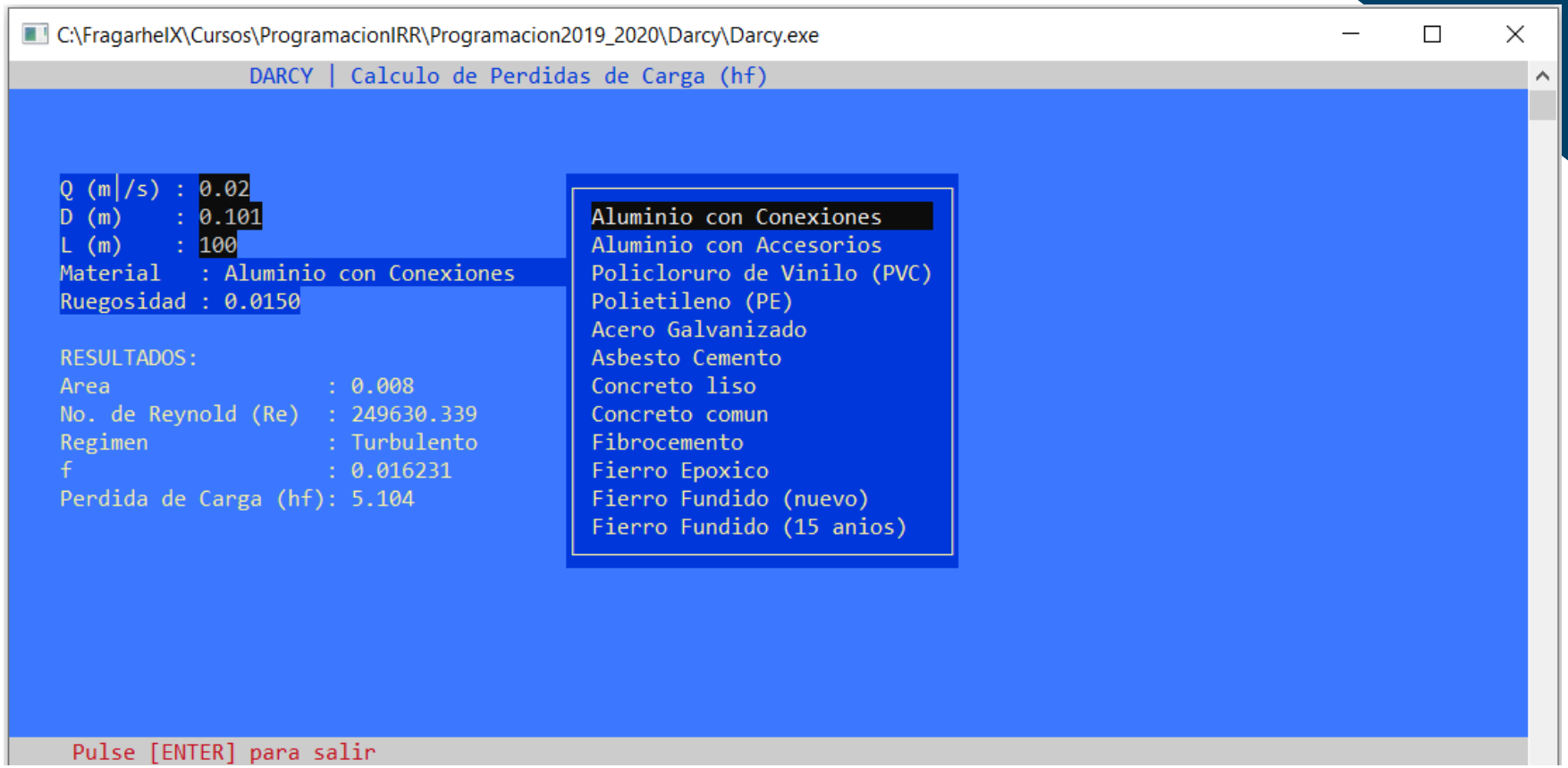
```
Function Colebrook_White(Re,Rr:Real):Real;
Const
   Tolerancia = 0.000001;
   NoIterMax = 100;
Var
  f0,f1 : Real;
  i     : Integer;

  Function f_G(Rr,Re,fp:Real) : Real;
  Begin
     // Expresion de Colebrook - White:
     f_G := power(1/(2*log10((Rr/3.71)+(2.51/(Re*sqrt(fp))))),2);
  End;

Begin
   f0:=64/Re;
   i:=0;
   While (i<=NoIterMax) do
   Begin
      f1:=f_G(Rr,Re,f0);
      if (Abs(f1-f0)<=Tolerancia) Then
         Begin
            Colebrook_White:=f1;
            Exit;
         End
      else
         Begin
            i:=i+1;
            f0:=f1;
         End;
   End;
   UbicaTexto(5,25,7,4,'El método falló, Cambie de método');
   Colebrook_White:=1;
End;
```





Pérdidas de carga en tuberías. Consola

```
// Cuerpo principal:
Procedure hf_DarcyWeisbach;
Const
   Vcinem = 0.00000101; {a T=20°C}
   Rug : Array[0..11] of Real =(0.0150,0.0150,0.0200,0.0020,0.0700,0.0750,
                                0.4000,0.9500,0.0250,0.0280,0.3125,0.5000);
Var
   Q,D,L,hf,v       : Real;    // Datos Entrada y/o Salida
   f,Re,A,Rr        : Real;    // Variables para el Calculo
   Rug_c            : Real;    // Coeficientes de Rugosidad
   Regimen          : String;
   Material         : Byte;

   Procedure Determina_f;
   Begin
      A:=(pi*Sqr(D))/4;
      v:=Q/A;
      Re:=(v*D)/Vcinem;
      Rr:=Rug_c/(D*1000);
      if (Re<=2000) Then    // Flujo Laminar
         Begin
            f:=64/Re; // Poseville
            Regimen:='Laminar';
         End;
      // Flujo Transicional
      if (Re>2000) And (Re<=4000) Then
         Begin
            Regimen:='Transicional';
            f:=Colebrook_White(Re,Rr);
         End;
      if (Re>4000) Then  // Flujo Turbuento
         Begin
            Regimen:='Turbulento';
            f:=Colebrook_White(Re,Rr);
         End;
   End;
```



Pérdidas de carga en tuberías. Consola

```
// Calculo de Perdidas de Carga:
Begin
   LimpiaPantalla(9,14);
   LimpiaLinea(1,7,14);
   LimpiaLinea(25,7,14);
   CentraTexto(1,7,1,' DARCY | Calculo de Perdidas de Carga (hf)');

 // Entrada de Datos:
   UbicaTexto(5,5,1,14,'Q (m³/s) : ');
   Readln(Q);
   UbicaTexto(5,6,1,14,'D (m)    : ');
   Readln(D);
   UbicaTexto(5,7,1,14,'L (m)    : ');
   Readln(L);
   Texto[1]:='Aluminio con Conexiones    ';
   Texto[2]:='Aluminio con Accesorios    ';
   Texto[3]:='Policloruro de Vinilo (PVC)';
   Texto[4]:='Polietileno (PE)           ';
   Texto[5]:='Acero Galvanizado          ';
   Texto[6]:='Asbesto Cemento            ';
   Texto[7]:='Concreto liso              ';
   Texto[8]:='Concreto comun             ';
   Texto[9]:='Fibrocemento               ';
   Texto[10]:='Fierro Epoxico             ';
   Texto[11]:='Fierro Fundido (nuevo)     ';
   Texto[12]:='Fierro Fundido (15 anios)  ';
   CuadroSimple(45,5,75,18,1,14);
   Material:=Selector(47,6,1,14,0,15,12);
   Rug_c:=Rug[Material-1];
   UbicaTexto(5,8,1,14,'Material   : '+Texto[Material]);
   UbicaTexto(5,9,1,14,'Ruegosidad : '+RealToStr(Rug_c,0,4));

   // Calculo de las Perdidas de Carga (hf)
   Determina_f;
   hf:=0.082626857*f*(power(Q,2)/power(D,5))*L;

   // Salida de Resultados:
   UbicaTexto(5,11,9,14,'RESULTADOS: ');
   UbicaReal(5,12,9,14,'Area                    : ',A);
   UbicaReal(5,13,9,14,'No. de Reynold (Re)   : ',Re);
   UbicaTexto(5,14,9,14,'Regimen               : '+Regimen);
   UbicaTexto(5,15,9,14,'f                     : '+RealToStr(f,0,6));
   UbicaReal(5,16,9,14,'Perdida de Carga (hf): ',hf);

   UbicaTexto(5,25,7,4,' Pulse [ENTER] para salir ');
   Readln;
End;

// Cuerpo Principal:
begin
   hf_DarcyWeisbach;
end.
```

Pérdidas de carga en tuberías. GUI

```
Public Function f_G(Rr As Single, Re As Single, fp As Double) As Double
f_G = 1 / (2 * Log10((Rr / 3.71) + (2.51 / (Re * Sqr(fp))))) ^ 2
End Function

Public Function Log10(x As Double) As Double
' logaritmo en base 10 (decimal)
    Log10 = Log(x) / Log(10#)
End Function
```

E16 =FF(C5,E7,C6,C8)

Ejemplo 2: Revisión/Calculo de Pérdidas de Carga

Datos:							
Q :	0.010	m³/s					
L:	120	m					
D:	3	in	0.0762	cm	76.2	mm	
e:	0.020	mm	*PVC*				
Vcine:	0.000001011	m²/s	*20°C*				
A:	0.004560367	m²					
v:	2.192805824	m/s					
e/D:	0.000262467						
Re:	165,273.79	*Turbulento*					
fo:	0.0003872362						
f:	0.0179232738		**0.0179233**				
hf:	**6.917433275**	m					

$$h_f = 0.08262686\, f \left(\frac{Q^2}{D^5}\right) L$$

$$f_{(i+1)} = \frac{1}{\left[2\log\left(\frac{\varepsilon/D}{3.71} + \frac{2.51}{Re\sqrt{f_{(i)}}}\right)\right]^2}$$

Calculo de f:

Tolerancia:	0.000001	
fo:	0.0003872362	Poiseville

Nolter	fo	f1	/ f1-fo /	Criterio
0	0.00038724	0.0264491356	0.02606189939830	
1	0.02644914	0.0174522005	0.00899693515132	
2	0.01745220	0.0179576133	0.00050541278496	
3	0.01795761	0.0179208310	0.00003678222969	
4	0.01792083	0.0179234623	0.00000263128391	
5	0.01792346	0.0179232738	0.00000018846668	Parar

Pérdidas de carga en tuberías. GUI-Excel

```
Public Function ff(Q As Single, D As Single, L As Single, e As Single) As Double
Dim Tolerancia As Single
Dim NoIterMax As Byte
Dim A As Single, v As Single
Dim Rr As Single, Re As Single
Dim Vcine As Single
Dim fo As Double, f1 As Double
Dim Salir As Boolean
Dim i As Integer

Const pi = 3.14159265358979
Vcine = 0.000001011
Tolerancia = 0.0000001
NoIterMax = 100

A = (pi * (D) ^ 2) / 4
v = Q / A
Re = (v * D) / Vcine
Rr = e / (D * 1000)
fo = 64 / Re    'Propuesta inicial: Poiseville
Salir = False
i = 0
Do
  f1 = f_G(Rr, Re, fo)
  If (Abs(f1 - fo) <= Tolerancia) Then
     MsgBox ("Se llego a la solucion en " & i & " iteraciones." & Chr(13) & "f = " & f1)
     ff = f1
     Salir = True
  Else
     i = i + 1
     fo = f1
  End If
Loop Until ((Salir = True) Or (i >= NoIterMax))
If (i >= 100) Then
   MsgBox ("El Metodo FALLO. Cambie de Metodo ")
   ff = -999.999
End If
End Function
```

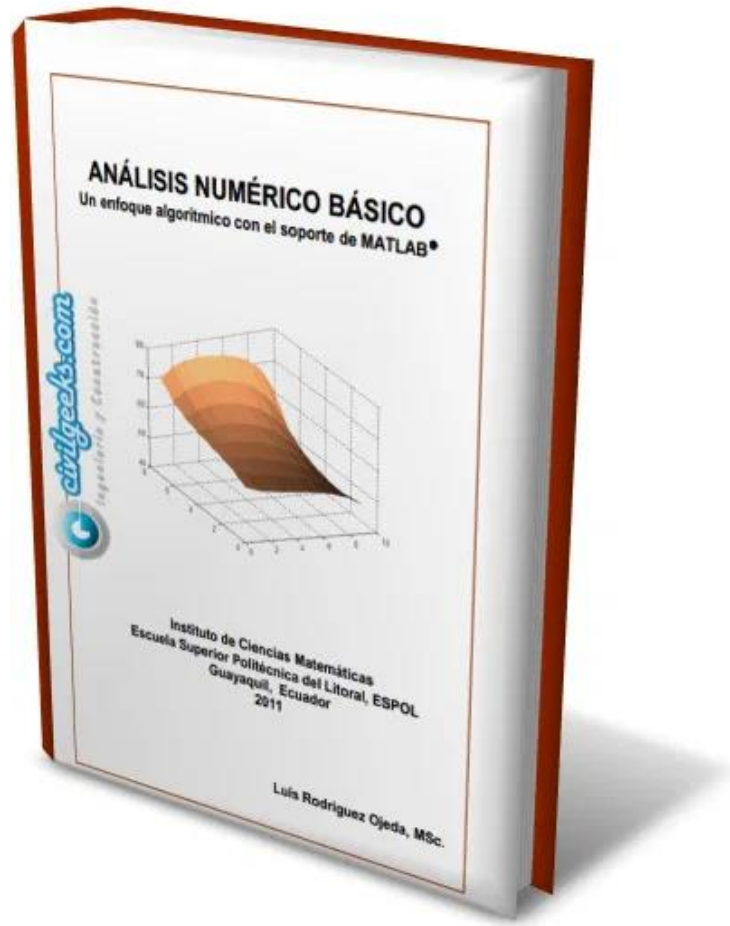
$$f_{(i+1)} = \frac{1}{\left[2\log\left(\frac{\varepsilon/D}{3.71} + \frac{2.51}{Re\sqrt{f_{(i)}}}\right)\right]^2}$$

4. Comentarios Finales

Capacitación para programadores

Además de un lenguaje de programación, para poder realizar una **aplicación robusta**, será importante para el programador, que se pueda capacitar en los siguientes rubros:

- Geometría analítica
- Estructura de Datos
- Análisis numérico

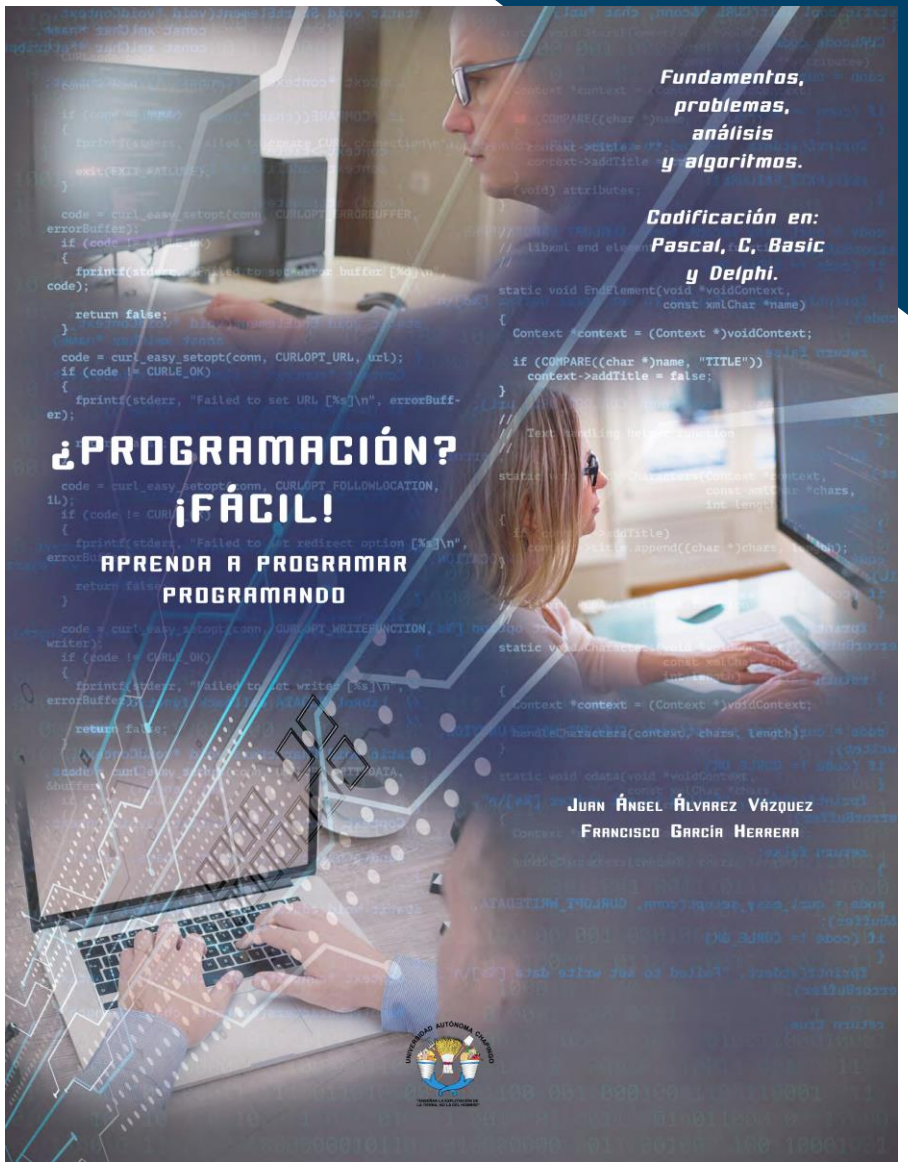


Comentarios Finales

El Ingeniero, debe integrar al grupo de herramientas que utiliza para el desarrollo de sus proyectos la ***PROGRAMACIÓN*** y el uso de un Lenguaje de Programación que le permita optimizar el tiempo de trabajo y aumentar su productividad.

Una Sugerencia …

Álvarez Vázquez, J.A., García Herrera, F. 2020. **¿Programación? ¡Fácil! Aprenda a Programar Programando**. UACh. Chapingo Edo. México.



Muchas gracias

Francisco García Herrera

Profesor–Investigador del Depto. de Irrigación

Universidad Autónoma Chapingo

fgarciah@chapingo.mx

Para citar esta presentación:

García Herrera, F. 2020. Los lenguajes de programación en la ingeniería: del aprendizaje a la aplicación. Serie de Seminarios Virtuales 2020. Colegio Mexicano de Ingenieros en Irrigación (COMEII). México. 56 pp.

Consulta el portal del COMEII y sus redes sociales:

www.comeii.com y **www.riego.mx**