UNIVERSIDAD DE LOS LAGOS

DIPLOMADO
LENGUAJE DE PROGRAMACIÓN C#

DIRIGIDO A:
DIRECCIÓN DE LOGÍSTICA DE CARABINEROS DE CHILE

JOEL TORRES CARRASCO
CRISTHIAN AGUILERA CARRASCO
CRISTIAN VALLEJOS VEGA

DEPARTAMENTO DE CIENCIAS DE LA INGENIERÍA
INGENIERÍA CIVIL EN INFORMÁTICA

Diferentes de Lenguajes de Programación

Index

The latest additions are shown in **bold**.

... قلب μλ

\# 1C-Enterprise 4D 4Test 8th

A ABAP4 ABC ACPI Source Language ACS Action! Actionscript (Flash 5) ActionScript (Flash 8) Actionscript (Flash MX) ActionScript 3.0 Ada ADVPL Agda aheui Algol-60 Algol-68 Alpha-Five-Xbasic Amazon States Language amharic Amiga-E AMOS Andl AngelScript Anguish Ansible Ante APC APL AppleScript Arduino Arena Arendelle Argh! ArnoldC AsciiDots ASP (C♯) ASP (JavaScript) ASP (VBE) ASP (VBS) ASP.NET Assembler (6502, Apple II) Assembler (6502, C64) Assembler (68000, Amiga) Assembler (68000, Atari ST) Assembler (68008) Assembler (8051) Assembler (ARM, Android) Assembler (ARM, RISC OS) Assembler (DG-Nova) Assembler (HLA) Assembler (HP-85) Assembler (IBM 370) Assembler (Intel) Assembler (Itanium) Assembler (MIPS) Assembler (MMIX) Assembler (PA-RISC) Assembler (PDP-11) Assembler (PDP-8) Assembler (PPC, Darwin) Assembler (SPARC) Assembler (TAS) Assembler (VP) Assembler (Win32) Assembler (X1) Assembler (X8) Assembler (z390) Assembler (Z80 Console) Assembler (Z80, CP/M) Assembler (ZX81) Asterisk ATS AutoHotkey AutoIT3 Automator AviSynth AWK

B B BAL BASIC-PICAXE BASIC Bato Batsh bc BCPL BeanShell Beef Befunge Beta BibTex Binary Lambda Calculus BIT BitZ Blender BlitzMax BlitzPlus BMC Remedy Boo Brainfuck BrightScript BS2000 BSP BuddyScript Buzz Byte Syze

C C* C/AL C♯ C (Amiga Anywhere) C (ANSI) C (Curses) C (GEM) C (Intuition) C (K&R) C (OpenGL) C (PresentationManager) C (Windows) C (X11 Athena) C++ C++ (.NET CLI) C++ (Epoc) C++ (FLTK) C++ (Gtk++) C++ (ISO) C++ (MFC) C++ (Qt) CA-Easytrieve Plus Caché Object Script CAML-Light Carp Casio BASIC CDuce Centura Ceylon ChaiScript Chef Chicken CICS-COBOL Clarion Clean Clipper Clojure CLP CMake COBOL Cobra Cocoa Coconut CoDScript CoffeeScript ColdFusion CommandScript Common Lisp Console Postscript ConTeXt Cool CoolBasic cpl Crystal CSS CUDA CWEB CYBOL

D D D4 Dafny Darkbasic Dart Databasic Dataflex dBase dc DCL Delphi Dialect DM DML Draco DWIM Dylan DynaMorph

E E easm Ecstasy Ecstatic Eiffel Elan ELENA 3.0 ELENA 4.0 Elixir Elliott Elm Emojicode Emoticon EOS 2 Erlang ERRE Eta Euphoria Eve Ezhil

F F# F Falcon FALSE Felix Fennel Ferite Fetlang Filemaker Fjölnir Flaming Thunder Focal FOCUS Folders Forth Fortran Fortran77 Fortran90 FortranIV Fortress FreeBASIC Frink Full Metal Jacket

G G-code Gambas GameMonkey Script Genero BDL Genie Gentee-simple Gentee GLBasic GML Go Gofer GoogleGadgets Gosu GRAMophone Gravity Gri Groovy Guile GynkoSoft

H Hack Harbour Haskell Have Haxe HDX HolyC Hoon HP-41C HP-48 HQ9+ HTML Hubot Human HyperTalk

I IBM-Exec IBM-Exec2 ici ICL SCL Icon IDC IDL Idris Imba Inform Informix 4GL Ingres ABF InstallScript Intercal Io Iptscrae

J J Jade Jako Janet Jason Java Java (Mobile) Java (Servlet) Java (Swing) Java Server Pages JavaScript JCL Jess Jorf JSFuck JudoScript Julia

K K K3 K4 K5 Kitten Kix Kotlin Kumir Kylix

L LÖVE L L33t LabVIEW Lasso LaTeX Latino LibertyBASIC Lily LilyPond Limbo LIMS Basic Lingo Linotte Lisaac Lisp-Emacs LiveCode LLVM Logo Logo (graphical) LOLCODE Loli LOTOS Lotus Note Formula Lotus Script LPC LS-DYNA LSL Lua

M m4 MACRO-10 MACRO-11 Macromedia-Flex MAD make Malbolge MAMASH Maple Marmelade Mathematica MATLAB Matrix MAX-MSP Maxima MAXScript MCSBL MDM Zinc MEL MetaPost MEX Microtik MiniScript mIRC-Alias mIRC-Commandline mIRC-Script MivaScript MML-AXE10 Modula-2 Modula-3 MoHAA-Script MOO Mouse MPD MPLAB IDE MS Small Basic MSDOS MSIL MuLisp Mumps MySQL FUNCTION Mythryl

N Natural Nemerle newLISP NewtonScript Nice Nim Nit Node.js Noor NSIS NXC

O Oberon.oberon Oberon.std Objective-C OCaml Occam Octave Omnimark Ook OpenVMS OPL.dialog OPL.simple Oz

P ParaSail Parser Pascal (Windows) Pascal Pawn PBASIC PDF PEARL PeopleCode Perl Perl 6 Pharo PHP+GD PHP Picat Piet Pike PILOT PL-SQL PL1 Plankalkül Pocket Calculator POP-11 Portugol PostgreSQL Postscript POV-Ray Powerbasic Powerflex PowerScript Powershell PPL PQN-PROC PRAAT ProC Processing Profan Prograph Progress Prolog Punyforth PureBasic (Console) PureBasic (Messagebox) PureBasic (Window) Python 2 Python 3

Q Q Qalb QCL qore QuakeC Quartz Composer QuickBASIC Quorum

R R Racket Rapira ratfor Rational Rose React-VR REALbasic RealText Reason Rebol-view REBOL Redcode REFAL-2 Refal Regular Expression Rexx (simple) Rexx (window) Rey rhine RPG IV v3-4 RPG IV v5 RPG IV v7.1 RPL RSL Ruby Rust

S S-Plus SAKO SAL SApp SAS Sather Sawzall Scala Scheme Scilab Scratch Seed7 Self SenseTalk Setl2 Shakespeare Shen SilverBasic SIMPLE Simula SinclairBasic Skip Smalltalk (simple) Smalltalk (window) Smalltalk MT SMIL SML Snobol Spin Spiral SPL4 Splunk SPL Spry SPSS SQL (Advantage) SQL (DB2) SQL (Oracle) SQL sqlplus SQR Squeak Squirrel SSI ST-Guide Stata Sucro SuperCollider SVG Swift SyMAL Symsyn szl

T T-SQL T T9 TACL TAL Tcl TECO TeX Texinfo Thue TI BASIC TI Extended BASIC TI-59 TI-8x TinyFugue Tk Toy Trans troff TSO CLIST Turing Machine Turing TypeScript

U Ubercode UniComal Unix Shell unlambda UnrealScript Ursala

V Vala var'aq **Vatical** VAX Macro VAX-11 Macro VBA (Excel) VBA (Word) VBScript Velato Velocity Verilog Vexi VHDL Vim script Visual Basic Visual Basic .NET Visual Basic 6 Visual FoxPro Visual Prolog VisualWorks Smalltalk VMS-DCL VRML VSL VVVV

W wenyan Whenever Whirl Whitespace wml Wolfram WSH

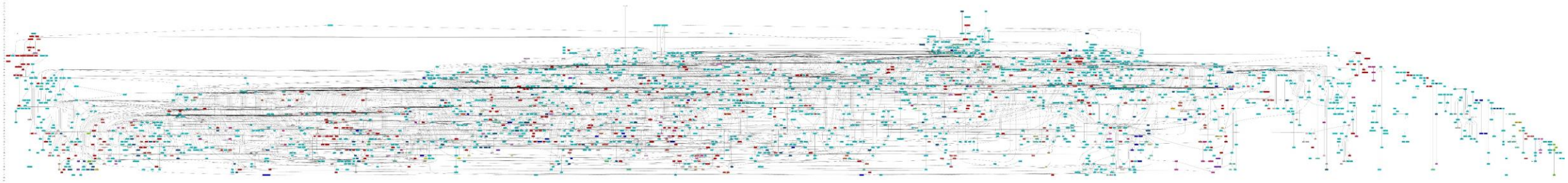
X X++ Xbase++ xblite XHTML XLogo XPL0 XQuery XSL-FO XSLT

Y Yacas

Z Zig ZIM

The Hello World Collection, disponible en: http://helloworldcollection.de/

Diferentes de Lenguajes de Programación



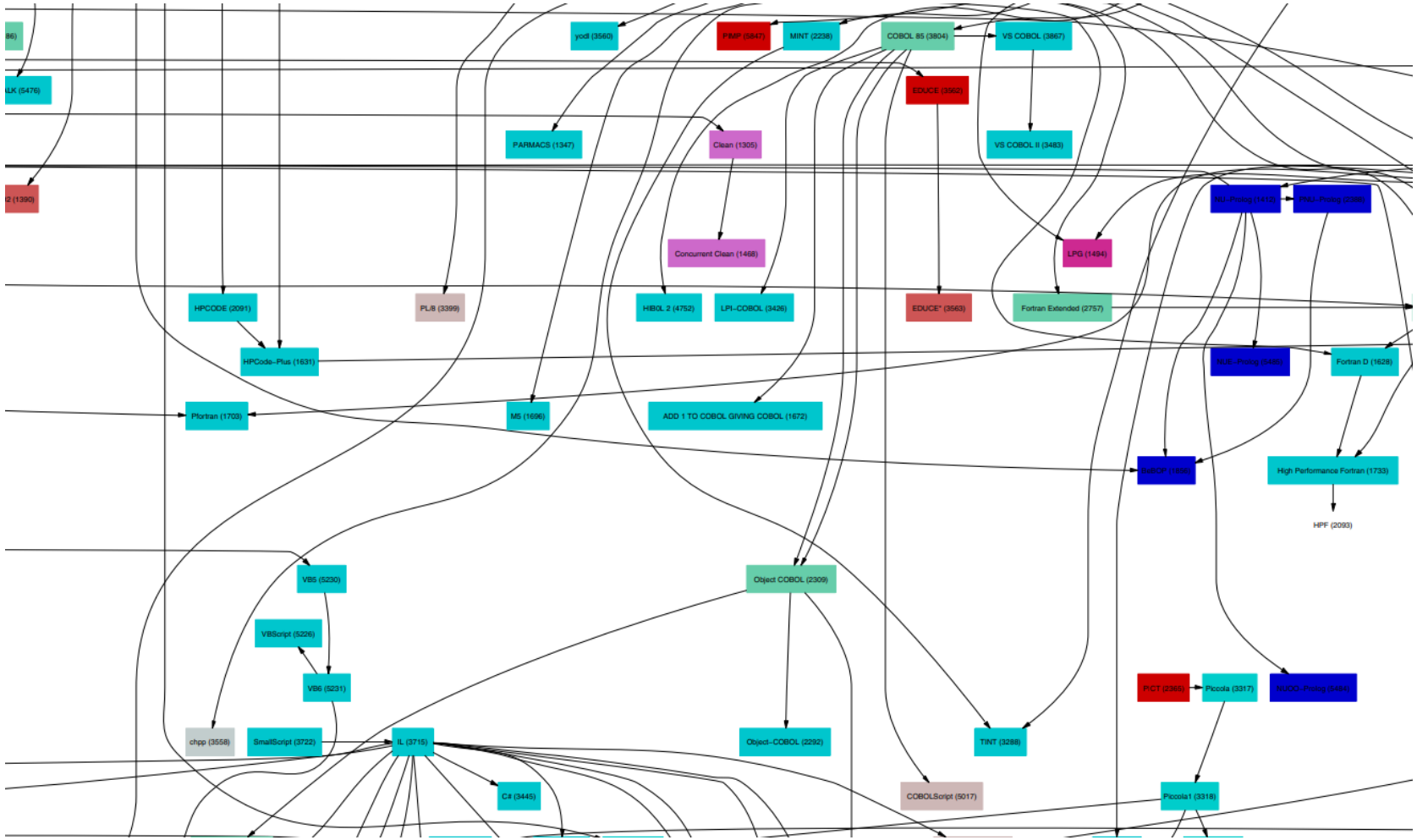
Online Historical Encyclopaedia of Programming Languages, Disponible en: https://hopl.info/

- Actualmente hay 8.945 lenguajes de programación desde el siglo XVIII
  - Cerca de 700 de ellos son todavía funcionales
  - Solamente 245 son relevantes

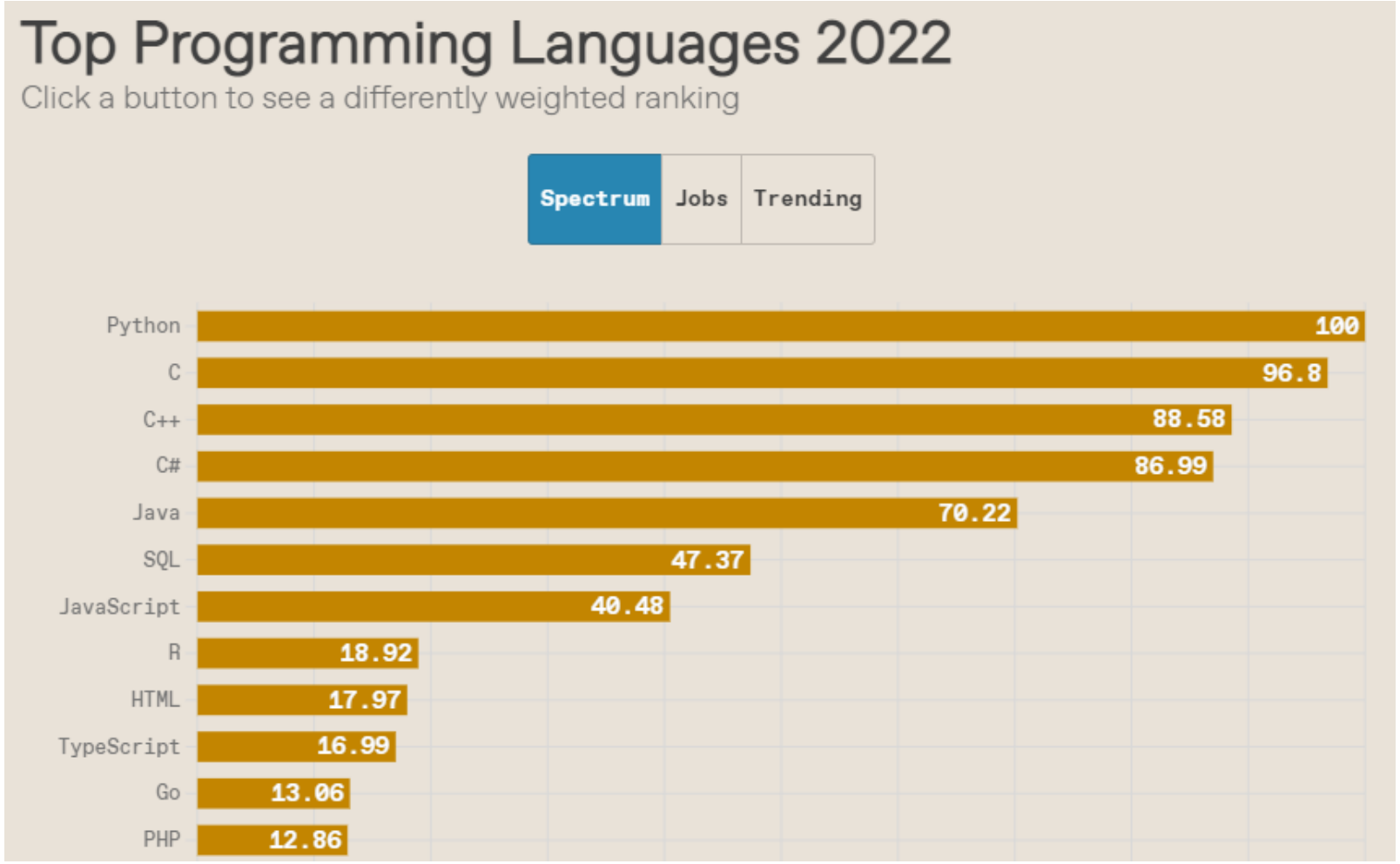
Artículo sobre la cantidad de lenguajes de programación, disponible en: https://devskiller.com/how-many-programming-languages/

Introducing The Pikachu Programming Language – A Programming Language Made For Pikachus, Disponible en: https://trove42.com/introducing-pikachu-programming-language/

Diferentes de Lenguajes de Programación



Ranking de Lenguajes de Programación



Ranking IEEE Spectrum, disponible en:
https://spectrum.ieee.org/top-programming-languages-2022##toggle-gdpr

Criterios para elegir un Lenguaje de Programación

- Aplicación
  - Existen lenguajes que están orientados a ciertas aplicaciones
- Salario
  - Hay lenguajes que permiten acceder a puestos mucho más competitivos
- Geografía
  - Cada país tiene un distinto nivel de avance tecnológico, por lo que se requieren ciertos lenguajes en específico
- Popularidad
  - Es más probable que se pida desarrollar en esos lenguajes
  - Además, la comunidad del lenguaje es más grande y robusta, por lo que permite la permanencia en el tiempo

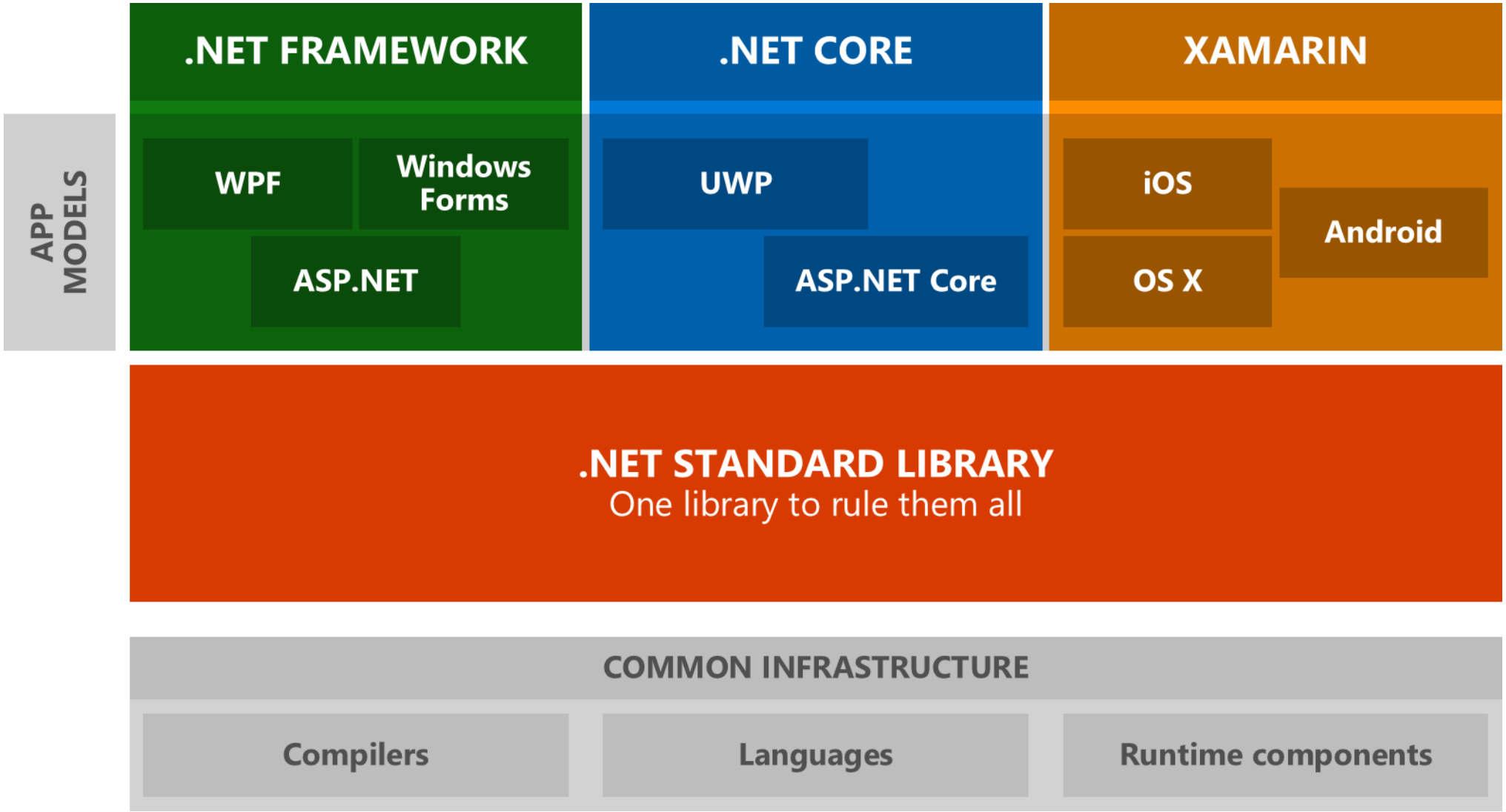
Introducción a C#

Introducción a C#

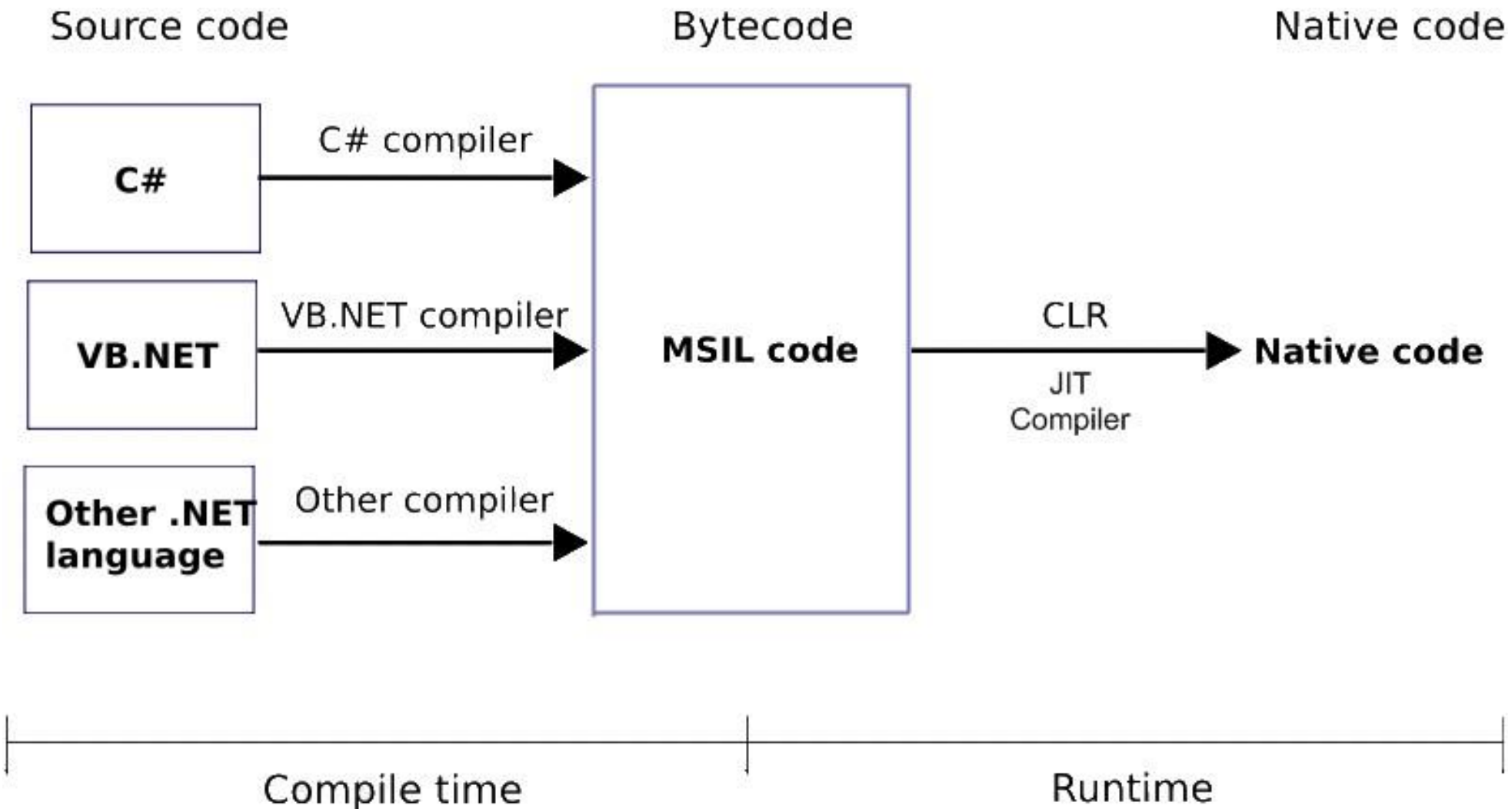
- Es un lenguaje de programación de código abierto desarrollado por Microsoft para que sea ejecutado sobre la plataforma **.NET Framework**
- Es considerado uno de los más usados en la industria
- Es de propósito general (sirve en entorno local, aplicaciones de escritorio, aplicaciones móviles, desarrollos web y videojuegos)
- Tiene una gran compatibilidad con toda la infraestructura del entorno de Microsoft

Documentación Oficial de C#, Disponible en: https://docs.microsoft.com/en-us/dotnet/csharp/

Introducción a C#



Introducción a C#



Traduciendo los elementos de algoritmos en los lenguajes de programación

Salidas de Datos

- Escribe texto en una línea cada vez que se ejecuta

```
Console.WriteLine("Hello World!");
```

- Escribe texto en la misma línea

```
Console.Write("Hello World!");
```

Entradas de Datos

- Recibe el texto que el usuario ha escrito en la consola:

```
string dato = Console.ReadLine();
```

Comentarios

- Comentario de una línea

```
//este es un comentario
```

- Comentario de Múltiples Líneas

```
/*este es un comentario
en multiples
lineas*/
```

Tipos de Datos

Data Type	Size	Description
int	4 bytes	Stores whole numbers from -2,147,483,648 to 2,147,483,647
long	8 bytes	Stores whole numbers from -9,223,372,036,854,775,808 to 9,223,372,036,854,775,807
float	4 bytes	Stores fractional numbers. Sufficient for storing 6 to 7 decimal digits
double	8 bytes	Stores fractional numbers. Sufficient for storing 15 decimal digits
bool	1 bit	Stores true or false values
char	2 bytes	Stores a single character/letter, surrounded by single quotes
string	2 bytes per character	Stores a sequence of characters, surrounded by double quotes

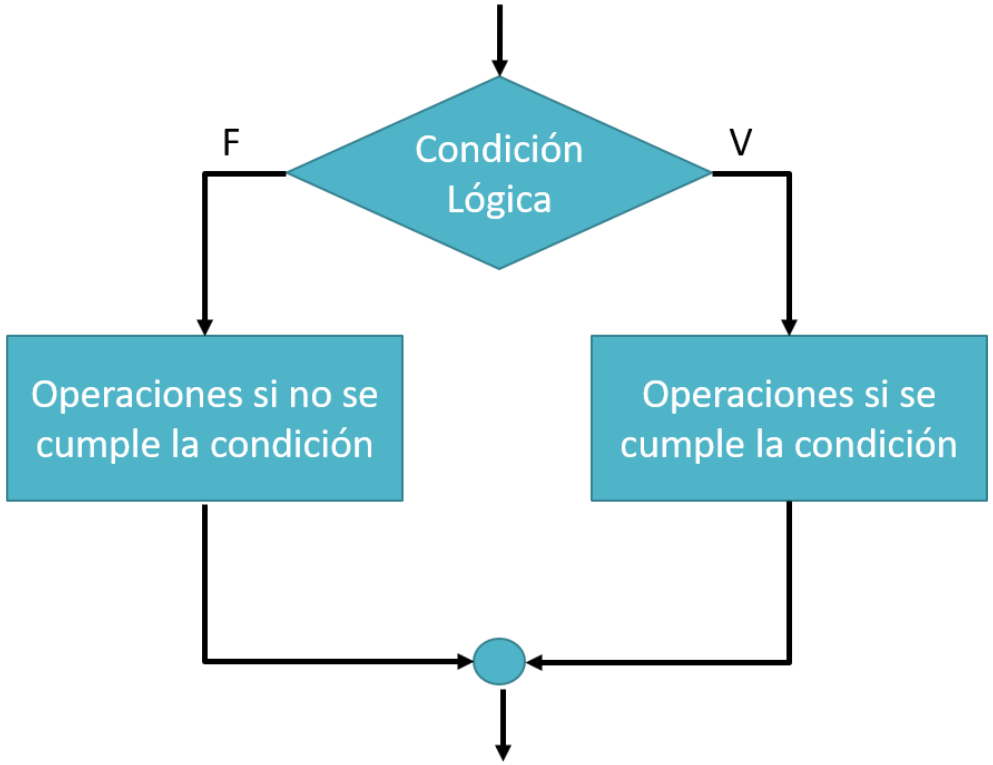
Declaración de Variables

- Requiere que cada variable sea declarada con anterioridad con su tipo de dato:

```
int myNum = 5;                // Integer (whole number)
double myDoubleNum = 5.99D;   // Floating point number
char myLetter = 'D';          // Character
bool myBool = true;           // Boolean
string myText = "Hello";      // String
```

Bifurcaciones: IF

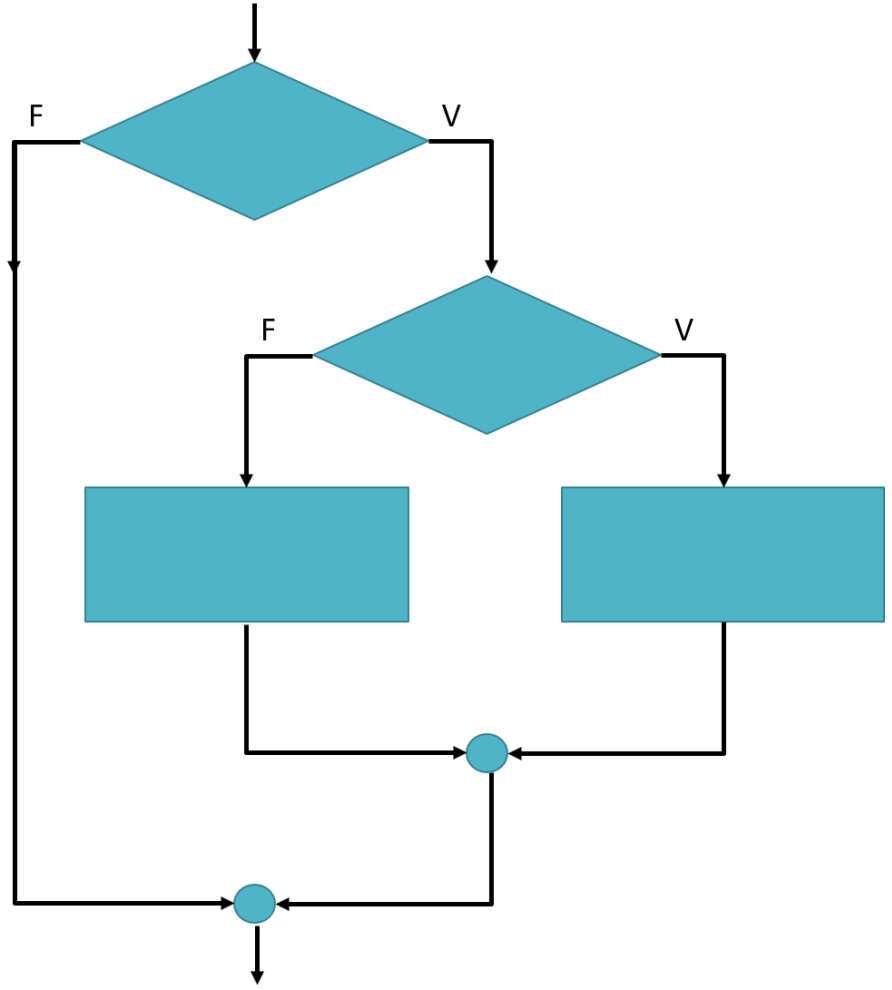
```
if (condición)
{
  // intrucciones si el código es verdadero
}
else
{
  // intrucciones si el código es falso
}
```



- Los límites están demarcados por las llaves {}

Bifurcaciones: IF anidados

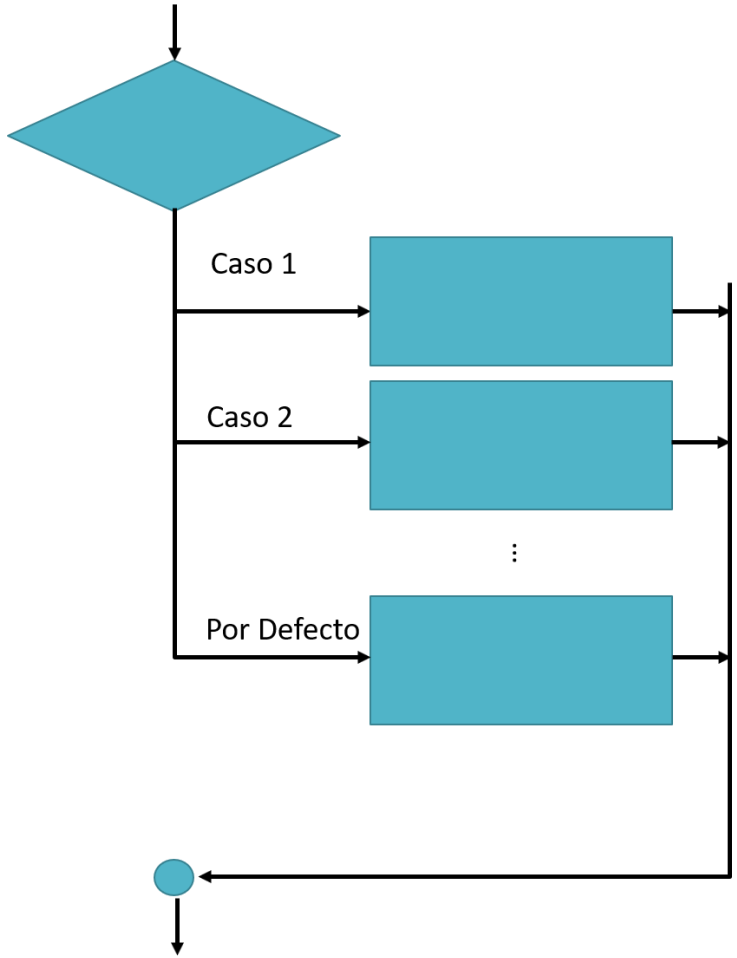
```
if (condición1)
{
  // instrucciones si la condición 1 es verdadera
}
else if (condición2)
{
  // instrucciones si la condición 2 es verdadera
}
else
{
  // instrucciones si ambas condiciones son falsas
}
```



Bifurcaciones: Switch

```
switch (variable)
{
  case valor1: // código si variable == valor1
    break;
  case valor2: // código si variable == valor2
    break;
  // se pueden describir todos los casos necesarios

  default: // código si no existe una caso detallado
           // es decir, el valor de la variable es
           // cualquier otro valor no detallado.
    break;
}
```



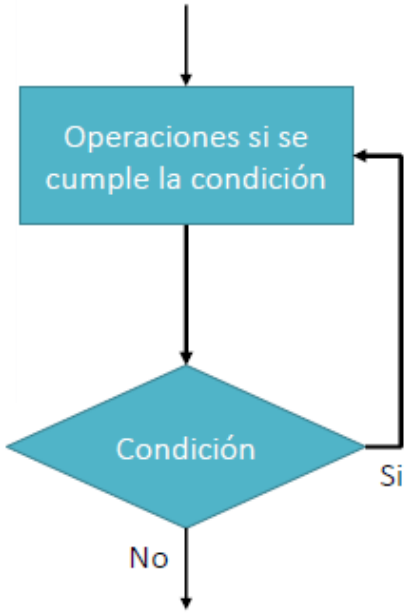
Ejercicio:

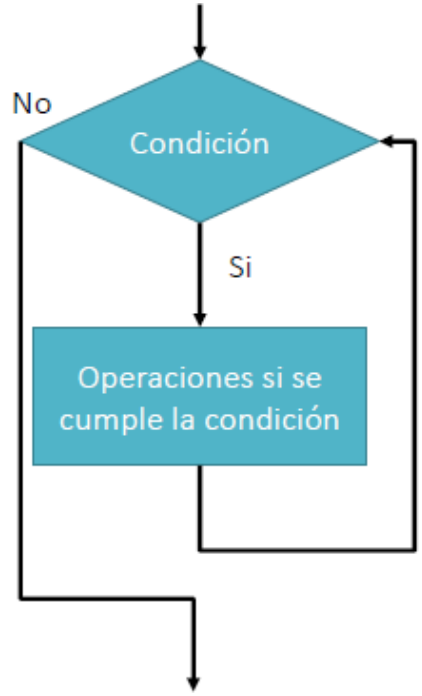
Saber si una persona es mayor o menor de edad

Ciclos: While

```
while (condición)
{
   // instrucciones cuando
   // la condición es verdadera
}
```

```
do{
   // instrucciones que se realizan
   // al menos una vez,
   // y se repite cuando la condición
   // es verdadera.
}while(consición)
```



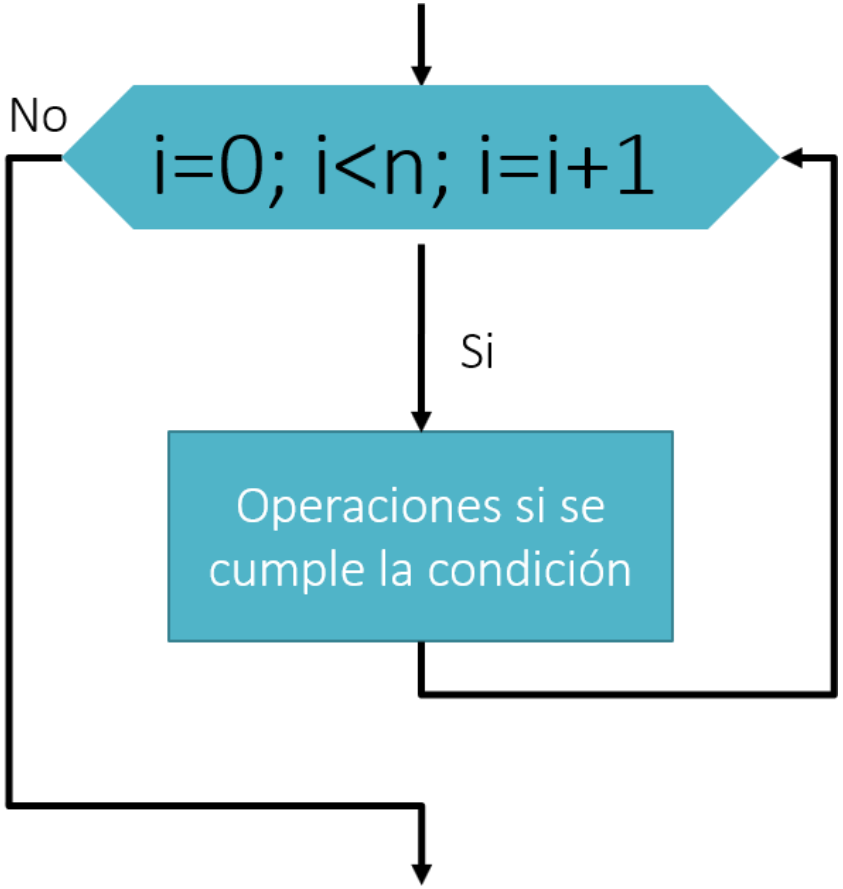


Ejercicio:
Cuenta regresiva de año nuevo desde 10 hasta 0

Ciclos: For

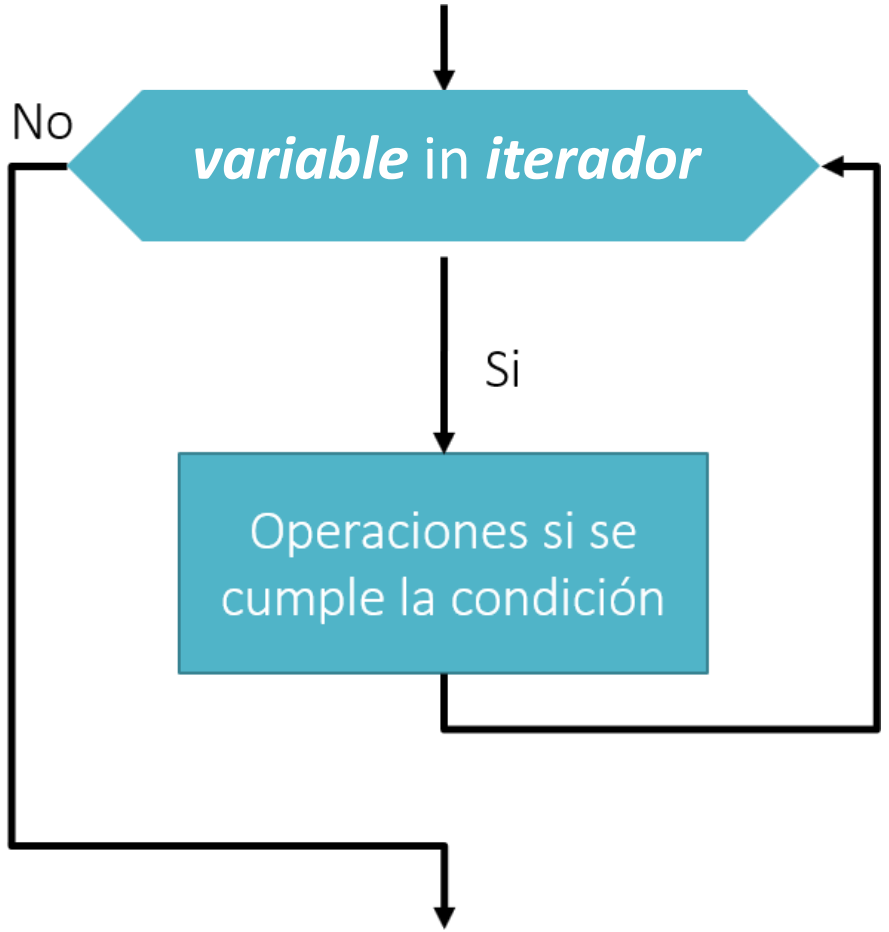
```
for (variable = valor_inicio; variable < valor_máximo ; variable = variable +1)
{
  // instrucciones cuando
  // la condición es verdadera
}
```

- Se compone de tres elementos
  - La inicialización
  - La Condición de Término
  - El paso de avance en el ciclo



Ciclos: Foreach

```
foreach (variable in iterador)
{
  // Instrucciones que se repiten
  // por cada elemento en el iterador
  // - iterador es una colección de datos (array)
  // - variable debe tener el mismo tipo de datos que el array
}
```



Ejercicio:
Cuenta regresiva de año nuevo desde 10 hasta 0

Ejercicio:
Obtener el inverso de un número

$$X^{-1}$$

Ejercicios

- Dado un valor entero **X**, calcular el perímetro y área de un círculo, cuyo diámetro es **X**.
- Desarrolle un algoritmo que permita calcular la siguiente función:

$$f(x) = (x+1)^2 + (2x)^2$$

- Desarrolle un algoritmo que permita calcular la distancia entre dos puntos del plano cartesiano

$$d = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$$

- Calcular la siguiente función:

$$|x| = \begin{cases} x & \text{si } x \geq 0 \\ -x & \text{si } x < 0 \end{cases}$$

Ejercicios

- Calcular las siguientes funciones:

A.

$$f(x) = \begin{cases} \frac{3xy+5y^2}{x-3} & \text{si } x > 2 \\ 3xy + 5x^2 & \text{si } x = 0 \\ 3xy + 5xy & \text{si } x < 2 \end{cases}$$

B.

$$f(a, b, c) = \begin{cases} a * b & \text{si } c > 1 \\ a/b & \text{si } c = 1 \\ a^2 * b^2 & \text{si } c < 1 \end{cases}$$

C.

$$f(x, y, z) = \begin{cases} 1 & \text{si } x < y/z \\ 0 & \text{si } x = y/z \\ -1 & \text{si } x < y/z \end{cases}$$

D.

$$f(x) = \begin{cases} x^2 - 2x - 5 & \text{si } x < 0 \\ x^3 - 3x + x2 & \text{si } x \geq 0 \text{ y } x < 3 \\ x^4 - 4x^3 + x^2 & \text{si } x \geq 3 \text{ y } x < 5 \\ x^5 - 5x^4 + x^3 & \text{si } x \geq 5 \end{cases}$$

Ejercicios

- Ingrese **N** números por teclado y que el algoritmo muestre el mayor, el menor y el promedio del conjunto de números.

- Construya un Algoritmo que calcule la Potencia

$$base^{exponente} = base * base * ...(exponenteveces)$$

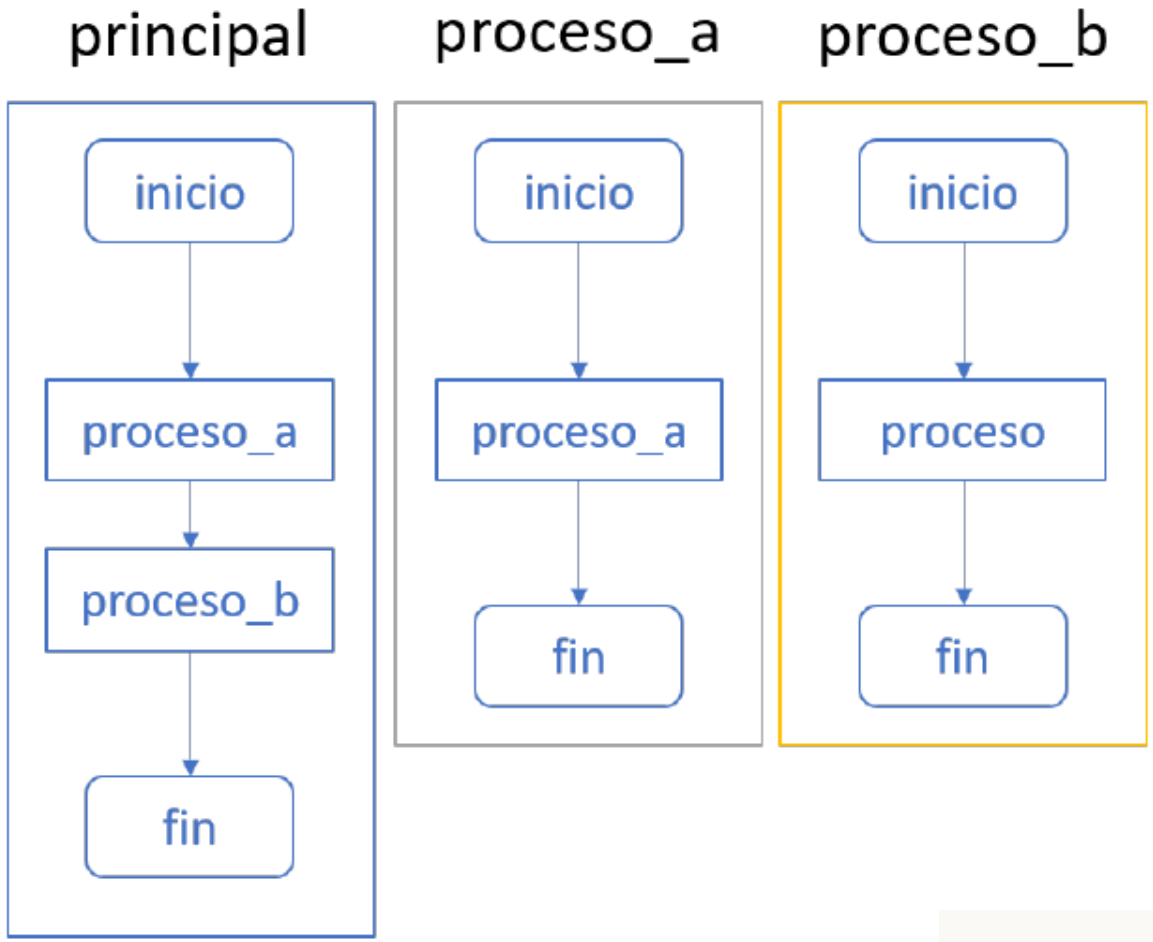
- Construya un Algoritmo que calcule el Factorial

$$valor! = valor * (valor - 1) * (valor - 2) * ... * 1$$

¿Qué son las funciones?

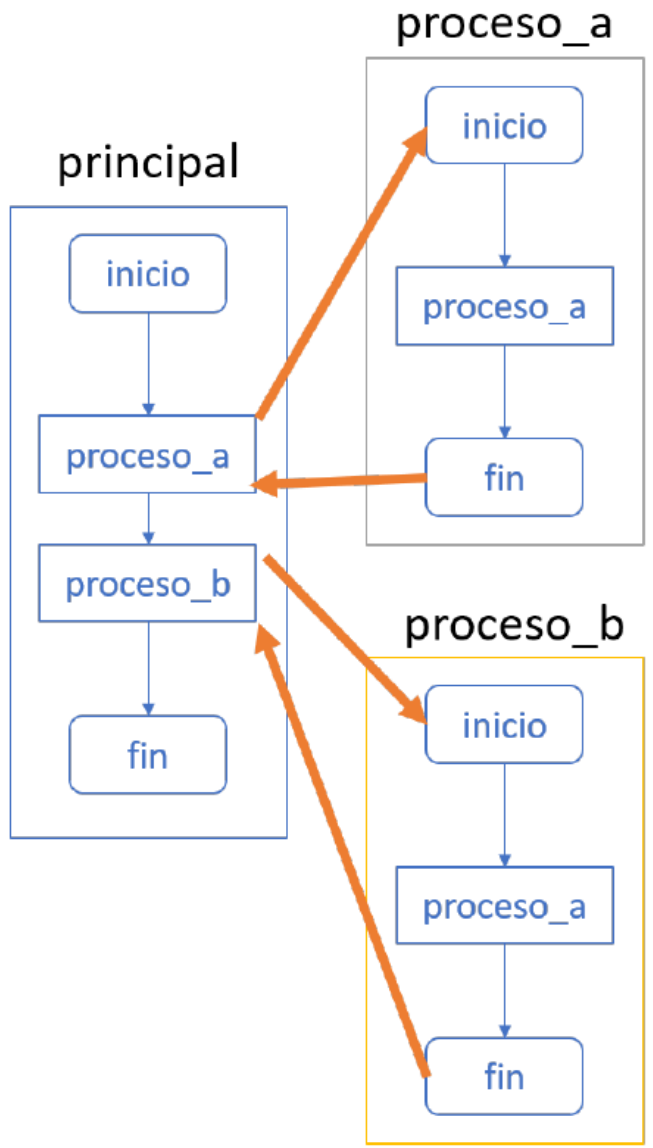
Las funciones son secciones de código o subrutinas.

- Agrupan un conjunto de instrucciones bajo el mismo nombre para realizar una tarea específica, donde puede (o no) retornar valor.
- Facilitan la resolución de problemas, dividiéndolo en pequeños subproblemas (Dividir para Conquistar).



¿Qué son las llamadas a las funciones?

- Una **llamada** a una función ejecuta las instrucciones de aquella función.
- Luego de terminar su ejecución, se **retorna** un valor si es necesario, y vuelve al punto donde se realizó la llamada.



Declaración de Variables

- Al igual que las variables, las funciones deben ser declaradas antes de ser usadas. (prototipado).
- Los parámetros son declaraciones de variables exclusivas para la función.
- La forma de declaración es la siguiente:

- Son llamadas Métodos estáticos
- Debe contener la palabra static, el tipo de datos, su nombre y los parámetros:

```
class Program{
  static tipoDato MiMetodo(tipoDato Argumentos){
    //Código a ejecutar
    //return (opcional)
  }
}
```

llamadas de Variables

- Las funciones pueden ser llamadas desde otras funciones o de la función principal *main*
- Pueden retornar valor a una variable o enviar directamente por consola.
- Si la función no tiene retorno, puede ser llamada directamente.

```
class Program{
  static void HolaMundo(){
    Console.WriteLine("Hola Mundo");
  }
  static void Main(string[] args){
    HolaMundo();|
  }
}
```

llamadas de Variables

- Las funciones pueden ser llamadas desde otras funciones o de la función principal *main*
- Pueden retornar valor a una variable o enviar directamente por consola.
- Si la función no tiene retorno, puede ser llamada directamente.

```
class Program{
  static void HolaMundo(){
    Console.WriteLine("Hola Mundo");
  }
  static void Main(string[] args){
    HolaMundo();|
  }
}
```

Ejercicio:
Función HolaMundo

Retorno de valor

- Si la función debe retornar un valor, lo hará usando la sentencia *return* dentro del cuerpo de la función, indicando el valor que retornará.
- El return obliga a la función a terminar. Omitiendo el resto del código.
- Las funciones que no tienen que retornar valor pueden omitir el return.

```
class Program{
  static int SumarUno(int numero){
    return numero+1;
  }
}
```

Ejercicio:
Función para sumar dos números enteros

Tipos de variables respecto a las funciones

Variable Local:

Es una variable que está dentro de una función.

Solamente puede ser usada dentro de dicha función.

Variable Global:

Es una variable que es declarada fuera de todas las funciones.

Puede ser usada en cualquier función.

Ejercicios

- Construya una función que reciba un String y lo muestre por pantalla
- Construya una función que permita recibir un número entero desde la consola
- Construya una función que reciba el valor de n (entero) y calcule la siguiente función:

$$S = \frac{1}{4} + \frac{3}{8} + \frac{5}{12} + \cdot + \frac{(2n+1)}{4n}$$

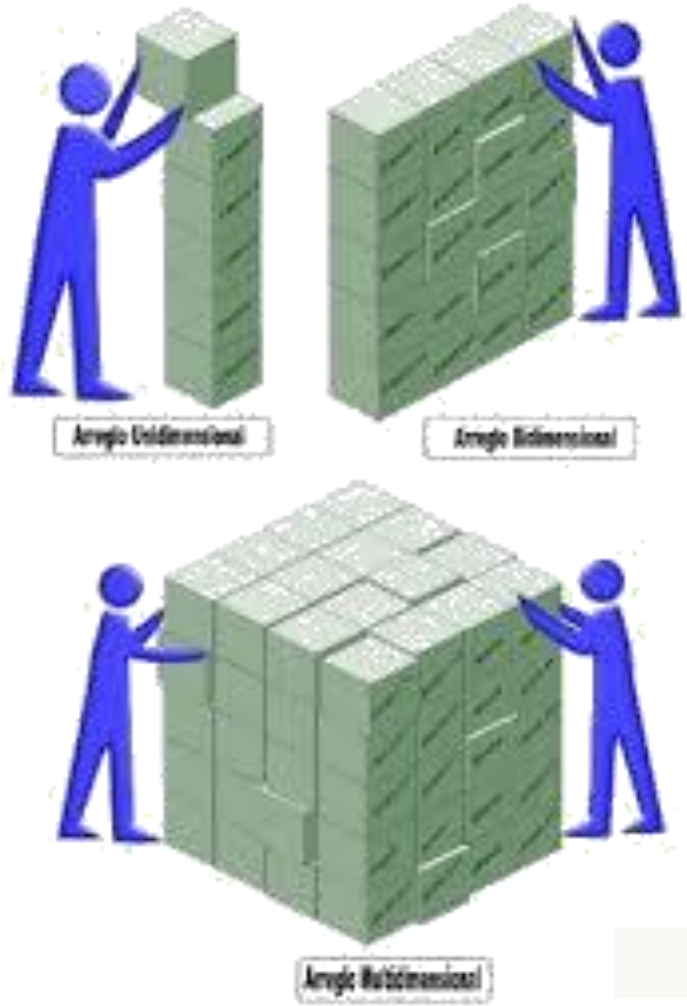
¿Qué son los Arreglos?

Los Arreglos son una colección consecutiva de variables del mismo tipo de dato, que tienen un nombre en común y pueden almacenar diversos valores.

Estos Arreglos ocupan un espacio de memoria consecutiva sin límite establecido. Sin embargo, la cantidad de posiciones está limitada al espacio disponible en la memoria.

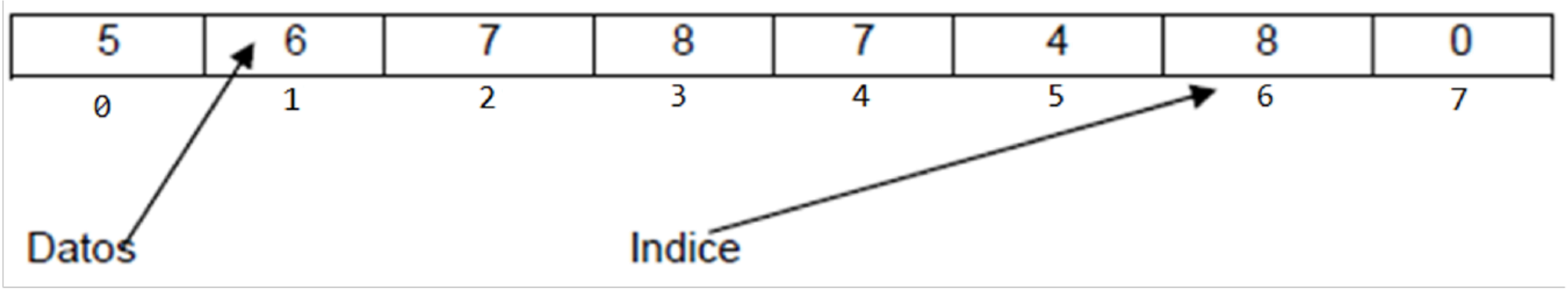
A estos tipos de arreglos se les conoce como arreglos estáticos.

Los arreglos estáticos deben ser declarados en el principio del programa con el fin de evitar desbordar la memoria.



Vectores

- Un Vector es un arreglo de una dimensión, que permite almacenar valores de un tipo de datos específico.
- La posición inicial de todo vector en programación es en cero (0).
- Y la posición final de un arreglo es dado por un total de posiciones N – 1.



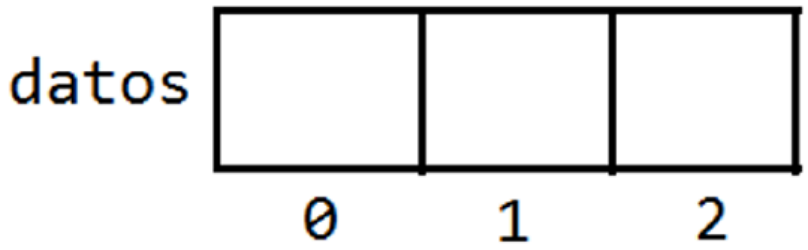
- Vector con 8 posiciones
- El índice indica la casilla correspondiente
- El dato puede ser cualquier valor según el tipo de dato correspondiente.

Implementación de Vector

```
int[] datos;
```

```
int[3] datos;
```

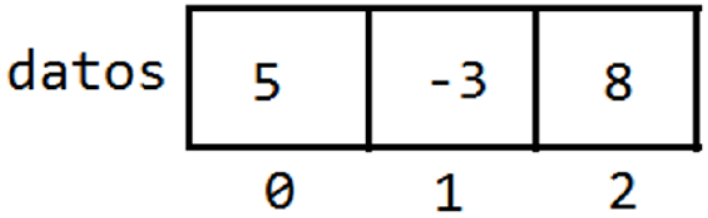
```
int[3] datos = {10, 20, 30};
```



Acceso a los datos del Vector

Lectura

```
Console.WriteLine(datos[0]);
Console.WriteLine(datos[1]);
Console.WriteLine(datos[2]);
```



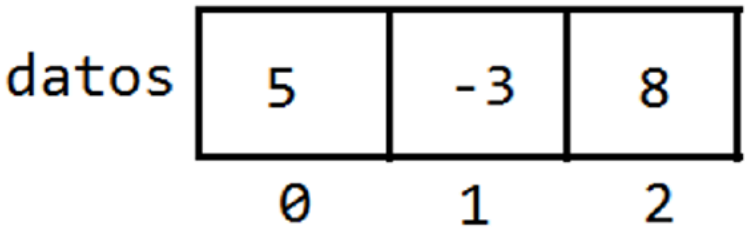
Escritura

```
datos[0] = Convert.ToInt32(Console.ReadLine());
datos[1] = Convert.ToInt32(Console.ReadLine());
datos[2] = Convert.ToInt32(Console.ReadLine());
```

Llenado de todas las posiciones de un Vector

```
using System;

namespace Arreglos
{
  class Program
  {
    static void Main(string[] args)
    {
      int i;
      int[3] datos;
      for(i = 0; i<3; i++)
        datos[i] = Convert.ToInt32(Console.ReadLine());
      for(i = 0; i<3; i++)
        Console.WriteLine(datos[i]);
    }
  }
}
```



Ejercicios

- Sumar todos los números de un vector de tamaño 8
- Mostrar los datos de un vector de manera inversa al índice almacenado
- Buscar el número menor y el número mayor de un vector de tamaño 7
- Mostrar la cantidad de números pares y la cantidad de números impares existentes en un vector de tamaño 15
- Ingresar hasta 300 números en un vector. Si el usuario ingresa un cero o un número negativo, el programa deja de solicitar números y muestra la suma de los índices pares.

¿Qué son las Cadenas?

Las cadenas de caracteres (también llamadas Strings) son un tipo particular de vectores, son de hecho vectores de tipo char.

Además el lenguaje nos permite escribirlas como texto dentro de comillas dobles. Sirven para almacenar palabras o frases, guardando solamente un caracter por espacio del vector, como se muestra en la figura:

0	1	2	3
H	o	l	a

Implementación de Cadena

```
string palabra = "Hola";
```

```
Console.WriteLine(palabra[0]);
```

0	1	2	3
H	o	l	a

Acciones sobre Cadenas

Acciones

- Identificación
- Tamaño
- Subcadenas
- Cambio de Formatos
- Índices
- Concatenación

```
using System;

namespace Arreglos
{
  class Program
  {
    static void Main(string[] args)
    {
      int indice;
      String[] palabra = "Hola";
      palabra[0];

      palabra.Lenght;


      if palabra.Contains("ol")
        Console.WriteLine("SI");

      indice = palabra.IndexOf("H");

      frase = palabra + "Como estás?"

    }
  }
}
```

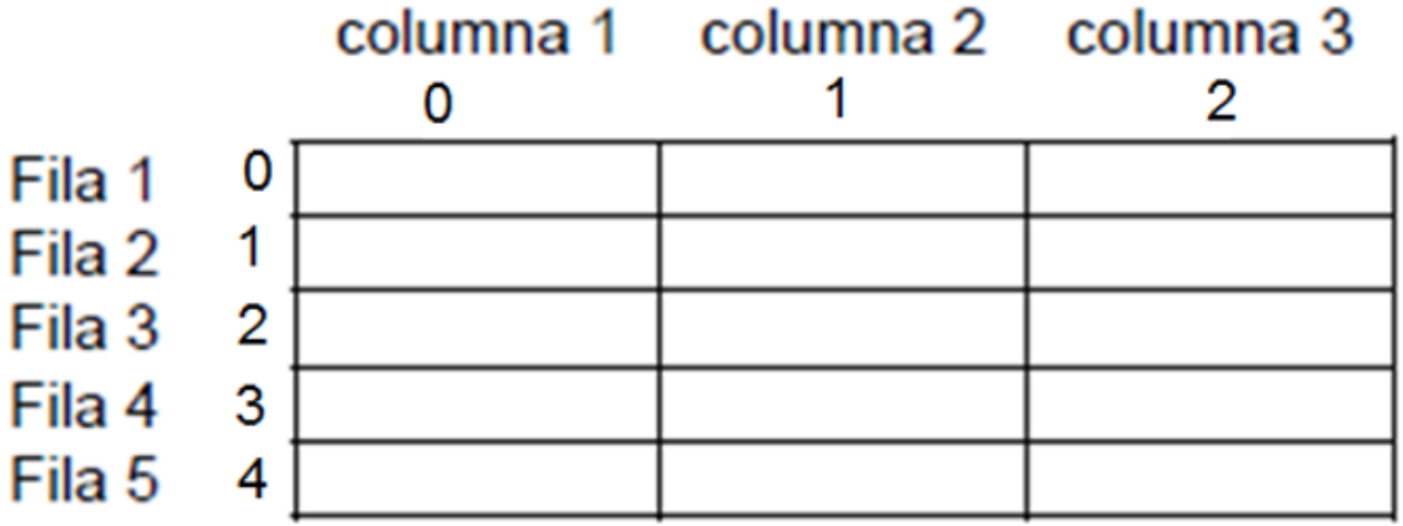
Ejercicios

- Escriba un programa que permita ingresar una palabra y cuente las letras que contiene.
- Escriba un programa que permita ingresar dos palabras y verifique cual es mayor (o si son iguales) en términos de cantidad de letras.
- Desarrolle un programa que permita identificar si dos palabras ingresadas por teclado son iguales. Muestre el resultado por pantalla.
- Desarrolle un programa que permita ingresar una palabra por teclado y luego muestre dicha palabra al revés.
- Desarrolle un programa que verifique si una palabra es Palindromo (Se caracteriza por leerse al revés y al derecho de la misma forma) ej: ’salas’
- Escriba un programa que permita ingresar los nombres de tres colores por el usuario y muéstrelos por pantalla.

¿Qué son las Matrices?

Las matrices, al igual que los vectores, son una colección de variables del mismo tipo, que tienen un nombre común (También son llamados Arreglos Multidimensionales).

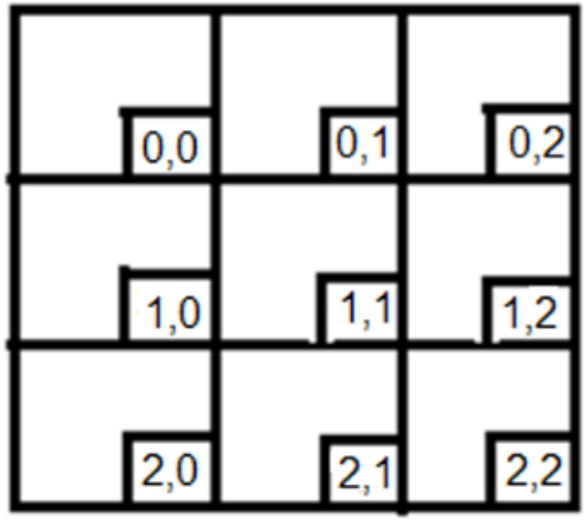
Al igual que en un vector, para acceder a una posición de una matriz se usa un índice. Sin embargo, este índice está compuesto por dos campos (x,y) o (filas, columnas).



Implementación de Matrices

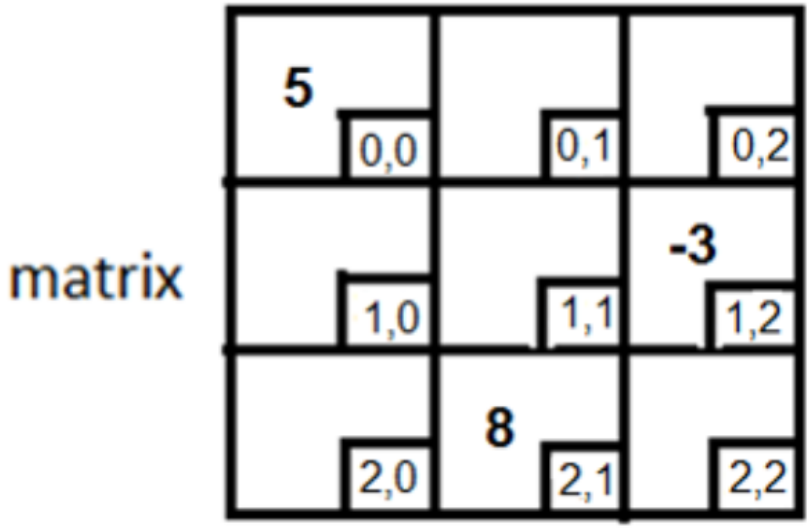
```
int[,] matriz;
matriz = new int[filas,columnas];
```

matrix



Acceder a las posiciones de las Matrices

```
matriz[0,0] = 5;
matriz[1,2] = -3;
matriz[2,1] = 8;
```



Ejercicios

Para cada ejercicio, crear un programa en C que permita:

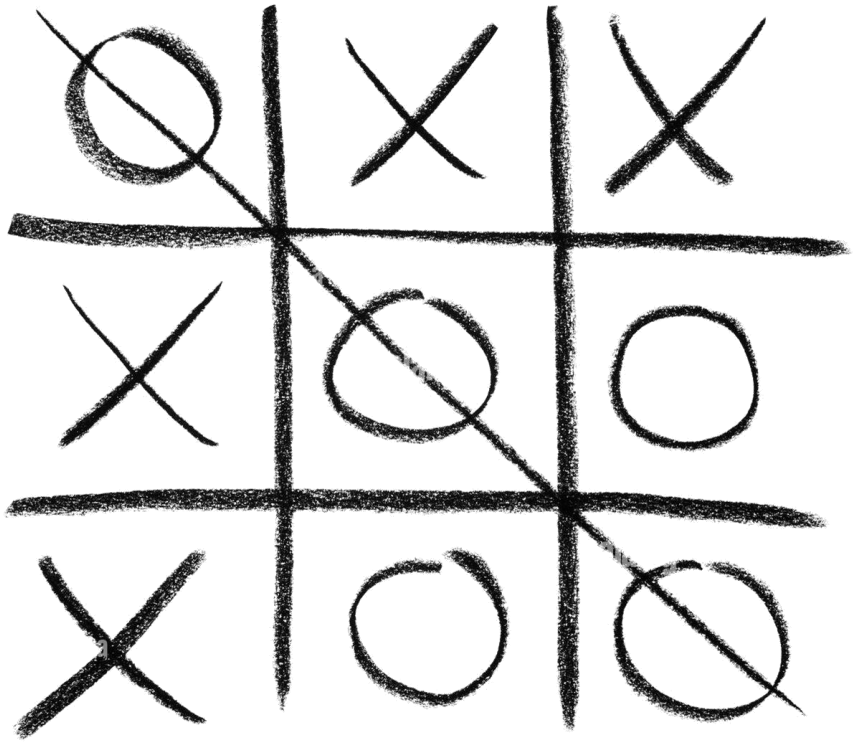
- Crear y llenar una matriz de 4x3 para luego sumar todos los datos. Mostrar el resultado por pantalla.
- Crear una matriz de 5x5 que contenga ceros (0) en las diagonales principales y los demás datos sean unos (1). Mostrar la matriz resultante por pantalla.
- Crear y llenar una matriz de 2x6 y contar la cantidad de números impares y números pares. Mostrar por pantalla ambos valores.

Ejercicio 1

- Desarrolle un programa que permita ingresar por teclado una fecha.
- Considere que la cadena debe ser validada según el formato dd-mm-aaaa o el formato dd/mm/aaaa.
- Además, la fecha tiene que ser coherente con el calendario, es decir:
  - no debe aceptar fechas que no existen; por ejemplo, 30-02-1999 NO EXISTE.
  - En caso de que la fecha ingresada no sea válida o no sea coherente, entonces el programa debe decir claramente por qué la fecha no es aceptada.

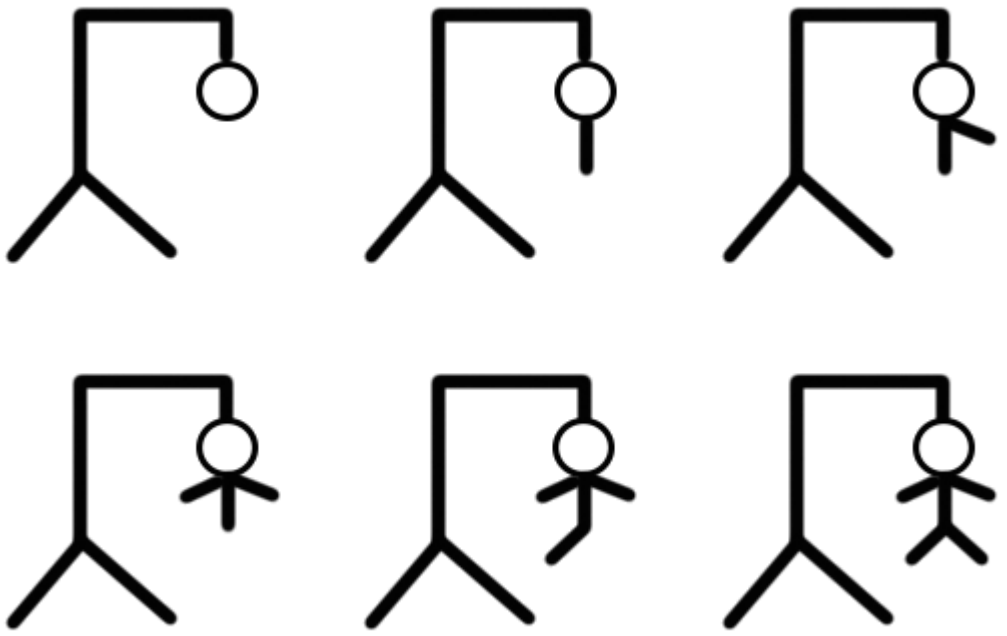
Ejercicio 2

- Desarrolle un programa que permita jugar al Gato.
- El Juego consiste en una grilla de 3x3 donde dos jugadores Player1 y Player2 (diferenciado por símbolos distintos), intentan formar una línea de tres de sus símbolos en línea horizontal, vertical o diagonal para ganar la partida, a través de turnos.
- Si ninguno de los jugadores han logrado realizar dicha línea cuando terminan todos los turnos, entonces el juego se considera un empate.
- Se pide:
  - Permitir el juego por
  - Imprimir el estado del tablero en cada turno
  - Determinar el ganador de la partida
  - Almacenar 5 partidas, identificando a los jugadores y al ganador de la partida
  - Imprimir lista de partidas



Ejercicio 2

- Desarrolle un programa que permita jugar El Ahorcado.
- El Juego consiste en ocultar una palabra de un jugador, entregándole solamente la cantidad de letras por los espacios vacíos de cada letra, y luego permitirle al jugador adivinar letras de la palabra o la misma palabra en una cantidad de intentos.
- La dinámica de juego permite entregar al menos una letra de la cadena como pista, luego cada 5 espacios es permitido mostrar una letra.
- Los intentos del jugador son dinámicos, debido a que debe mantener a salvo a un sujeto imaginario, por cada intento fallido se elimina una extremidad del sujeto, teniendo un total de 6 intentos fallidos antes de perder el juego.
- Si por el contrario, en un intento acierta a una de las letras, no se cuenta aquel intento, manteniendo a salvo al sujeto.
- Finalmente, si el jugador intenta adivinar la palabra completa, entonces tiene dos opciones, independiente de la cantidad de intentos, debe coincidir la palabra ingresada con la palabra original, o pierde ya que no ha acertado.
- Se pide:
  - Permitir ingresar la palabra original y mantenerla oculta del jugador.
  - Mostrar en todo momento el estado del sujeto imaginario
  - Mostrar los espacios y algunas letras correspondientes a la palabra original
  - Identificar si el jugador a terminado sus intentos y determinar el término del juego
  - Identificar si el Jugador ha ganado el Juego



Preguntas

UNIVERSIDAD DE LOS LAGOS

DIPLOMADO
LENGUAJE DE PROGRAMACIÓN C#

DIRIGIDO A:
DIRECCIÓN DE LOGÍSTICA DE CARABINEROS DE CHILE

JOEL TORRES CARRASCO
CRISTHIAN AGUILERA CARRASCO
CRISTIAN VALLEJOS VEGA

DEPARTAMENTO DE CIENCIAS DE LA INGENIERÍA
INGENIERÍA CIVIL EN INFORMÁTICA