



1859



Universidad  
Nacional  
de Loja

*Educamos para* **Transformar**





Universidad  
Nacional  
de Loja

# Facultad de la Energía, las Industrias y los Recursos Naturales no Renovables

## Computación

### Fundamentos de Algoritmos y programas

1 ciclo A

Teoría de la Programación

Unidad 1

Septiembre 25 - Febrero 26

*Educamos para* **Transformar**



Universidad  
Nacional  
de Loja

# Fundamentos de Algoritmos y programas



## Contenido

- Programación en C
- Programación en Java
- Programación en Python



## { Codificación

Realizar la conversión del algoritmo a lenguaje de programación implica que no solo se **sustituyan** las palabras reservadas en español por sus homónimos en inglés, sino también que se agreguen detalles como la declaración de variables, constantes y **librerías** que son omitidas por el pseudocódigo.

Inclusión de librerías
Declaración de constantes
Comienzo de programa
Declaración de variables
Cuerpo del programa

El código queda almacenada en lo que llamamos **programa fuente**, y el Lenguaje C utiliza archivos con **extensión .c**, es decir **nombre\_archivo.c** }

## { Librerías o Bibliotecas

Son archivos que se encuentran en la cabecera de los programas. Las librerías contienen código, para realizar operaciones y cálculos de uso frecuente y son parte de cada compilador.

El programador debe invocar todos aquellos librerías o bibliotecas que necesite.

La extensión de un archivo de librería es **.h**, es decir nombre\_librería.h

```
#include <librería.h>
```





## { Librerías o Bibliotecas

La librería que nunca puede faltar es **stdio.h** que contiene los prototipos de funciones y los tipos de datos para manipular sus entradas y salidas.

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <math.h>
#include <time.h>
...
```



## { Comienzo del programa

En el pseudocódigo el algoritmo tenía un inicio y un fin, en codificación será:

```
1  #include <stdio.h>
2
3  int main(){
4      |
5      return 0;
6  }
```

El próximo paso será **compilar** el **código fuente** para obtener el **programa ejecutable** y poder ejecutarlo (correrlo) en la computadora.

En C el programa empieza SIEMPRE desde la función **main()**.

}



## Windows por terminal:

- Compilarlo:
  - `gcc .\hola_mundo.c -o hola_mundo`
- Ejecutar
  - `.\hola_mundo.exe`

## Linux (Ubuntu/Debian) por terminal:

- Compilarlo:
  - `gcc hola_mundo.c -o hola_mundo`
- Ejecutar
  - `./hola_mundo`

El **compilador** traduce directamente el código a **lenguaje máquina** (código binario).

- El resultado es un **.exe** en Windows y en Linux sin extensión (**ejecutable**).
- El archivo corre directamente en el Sistema Operativo (SO) sin necesitar otro programa.





Universidad  
Nacional  
de Loja

# Instalación del Lenguaje C

The screenshot displays the Visual Studio Code interface. On the left, the Explorer pane shows a file named `hola_mundo.c` and its executable `hola_mundo.exe`. The main editor area shows the source code of `hola_mundo.c`:

```
C hola_mundo.c > main()
1  #include <stdio.h>
2  int main() {
3      printf("Hola, mundo en C desde VS Code!\n");
4      return 0;
5  }
```

Below the editor, the TERMINAL pane shows the compilation and execution commands:

```
PS C:\Users\NIS\Documents\Programacion\C> gcc hola_mundo.c -o hola_mundo
PS C:\Users\NIS\Documents\Programacion\C> .\hola_mundo.exe
Hola, mundo en C desde VS Code!
PS C:\Users\NIS\Documents\Programacion\C> 
```

`\n`

```
Símbolo del sistema

Microsoft Windows [Versión 6.3.9600]
(c) 2013 Microsoft Corporation. Todos los derechos reservados.

C:\Users\LIS>cd C:\Users\LIS\Documents\Programacion\C

C:\Users\LIS\Documents\Programacion\C>dir
El volumen de la unidad C no tiene etiqueta.
El número de serie del volumen es: 3632-C730

Directorio de C:\Users\LIS\Documents\Programacion\C

06/10/2025  09:15    <DIR>          .
06/10/2025  09:15    <DIR>          ..
06/10/2025  09:11             100 hola_mundo.c
06/10/2025  09:15        40.764 hola_mundo.exe
                2 archivos             40.864 bytes
                2 dirs   365.138.006.016 bytes libres

C:\Users\LIS\Documents\Programacion\C>gcc hola_mundo.c -o hola_mundo

C:\Users\LIS\Documents\Programacion\C>hola_mundo.exe
Hola, mundo en C desde VS Code!

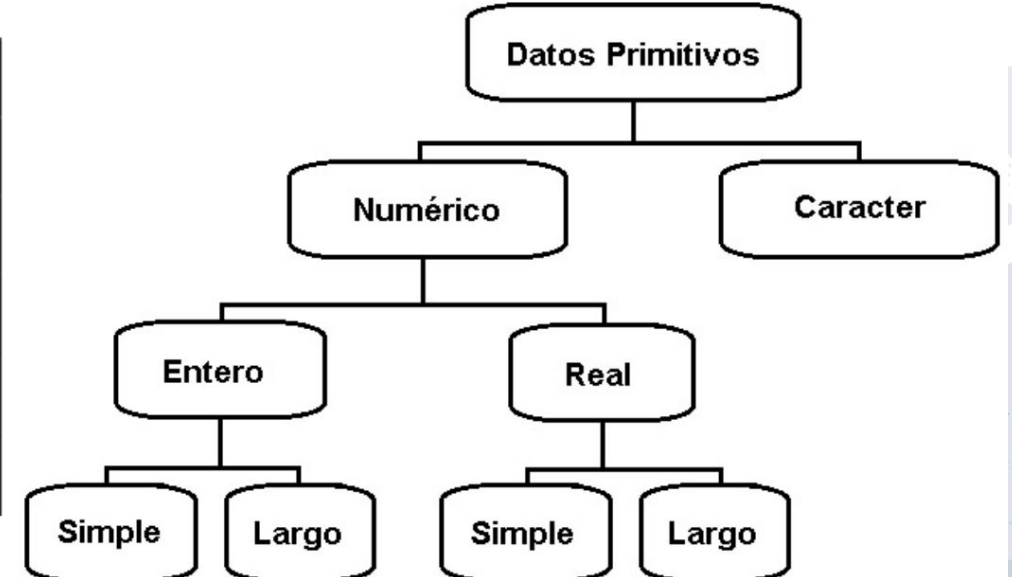
C:\Users\LIS\Documents\Programacion\C>
```



## { Tipos de datos simples

El tipo de dato determina la cantidad de memoria para almacenar (Guerra Salazar et al., 2023).

TIPO	ANCHO EN BIT	RANGO EN PC
char	8	-128 a 127
Int	16	-32768 a 32767
long	32	-2147483648 a 2147483647
float	32	3.4E-38 a 3.4E+38
double	64	1.7E-308 a 1.7E+308



## { Variables

Las variables se declaran al inicio del programa, y antes de que se utilicen en las operaciones.

- `tipoDato nombVariable; /*Declaración de una variable */`
- `tipoDato nomVar1, nomVar2; /*Declaración de dos variables del mismo tipo de dato */`
- `tipoDato nomVar = valor_inicial; /*Declaración e inicialización de una variable */`

}



## { Variables

- `int a, b;`
- `int edad, cont=0;`
- `int metrosLargo, metrosAncho;`
- `float acumulador=0;`
- `char respuesta, opcion='S';`



## { Variables

Siempre culminan con un `'\0'` que representa la marca del fin de la cadena (no es un **símbolo visible**).

**Cadena.** En el **lenguaje C** se trata de casos especiales de **vectores** de tipo **char** (carácter).

`char cadena1[]="a";` equivale a `char cadena1[]={ 'a', '\0' }`

**NOTA:** `char cadena[4]="hola";` A veces parece funcionar, pero puede sobrescribir memoria vecina, mostrar basura o hasta provocar un fallo `cadena[5]`.



## { Variables

- `char nombre[10] = "Maria";`
- `char nombre[] = "Maria";`
- `* nombre = "Maria";`

[Video1](#)

[Video2](#)

}

## { Constantes

Pueden declararse globalmente, es decir, fuera de las funciones que componen el programa, o localmente, es decir, dentro de una determinada función.

### Constantes tipadas

- `const float IVA15 = 0.15;`
- `const float PI = 3.141597`
- `const char UNIVERSIDAD[] = "ESPOCH";`





## { Constantes

- `#define nombre_constante valor_constante`
- `#define MAX 100`
- `#define PI 3.1416`

**Atención:** la declaración de constantes no necesita ; (punto y coma) para finalizar la línea de comando.

}

## { Asignación

Se utiliza el signo igual **(=)** como símbolo de asignación.

- `a = a + 5;`
- `suelo = 450;`
- `estadoCivil = 'D';`

}

## { Asignación

También podemos combinar otros operadores de asignación que el lenguaje permite.

- $a += b;$

Equivale :  $a = a + b;$

- $a -= b;$

Equivale :  $a = a - b;$

- $a *= b;$

Equivale :  $a = a * b;$

- $a /= b;$

Equivale :  $a = a / b;$

- $a \% = b;$

Equivale :  $a = a \% b;$

}



## { Asignación

Los operadores **++** (**incremento en 1**) y **--** (**decremento en 1**).

`b++;`  
`++b;`  $\Rightarrow$  `b=b+1`

```
int b = 5;  
b++;    // ahora b = 6  
++b;    // ahora b = 7
```

`b--;`  
`--b;`  $\Rightarrow$  `b=b-1`

```
int b = 5;  
b--;    // ahora b = 4  
--b;    // ahora b = 3
```

}

## { Asignación

Caso 1:

```
b=16;
```

```
y=++b;
```

Pre-incremento:

- Primero se incrementa b en 1.
- Luego se asigna el nuevo valor a y.
- `b = 17; y = 17`

Caso 2:

```
b=16;
```

```
y=b++;
```

Post-incremento:

- Primero se asigna el valor de b a y.
- Luego b se incrementa en 1.
- `y = 16; b = 17`



## { Máscaras más comunes

Máscara	Imprime
%i	Un entero
%d	Un entero
%c	Un único carácter
%s	Una cadena de caracteres
%(número)s	Una cadena de caracteres limitada por un número, por ejemplo: %5s (en estos casos imprimirá los cinco primeros caracteres)
%%	% Esto es por si queremos imprimir el símbolo de porcentaje
%(número1).(número2)f	Un número con decimales. El tamaño es número1 y la cantidad de decimales es número2. Por ejemplo: %6.2f (el tamaño del número será 6 y tendrá 2 decimales)

- **%g:** El dato se considera de tipo float.
- **%f:** El dato se considera de tipo float.
- **%lf:** El dato se considera de tipo double.
- **%li:** El dato es un long int (rango más amplio que el int normal).
- ...



{ Entrada de datos (lectura)      Ingresar datos por el usuario.

Leer → **scanf**

`scanf (“cadena_control_tipo”, &variable);`

Tiene asociados dos parámetros:

- La **máscara**, formato que indica el tipo de dato esperado.
- **Variable** de ingreso, delante de la variable hay un «&» que significa la dirección de la variable, donde se va a almacenar el dato.

**&** para *Enteros, decimales y caracteres*.

}

## { Entrada de datos (lectura)

- `scanf("%d", &ed);`
  - `scanf("%f", &r);`
  - `scanf("%c", &sal);`
  - `scanf("%i %i", &a, &b);`
  - `scanf("%s", nombres);` Lee hasta encontrar un espacio en blanco.
  - `scanf("%4s", nombres);` Lee 4 caracteres hasta encontrar un espacio en blanco.
- Atención:** En cadenas, el `scanf` define la lectura hasta el primer espacio en blanco.
- `scanf("%[^\\n]", nombres)`
  - `scanf("%19[^\\n]", nombres);` Lee 19 caracteres hasta encontrar un salto de línea.

{ Salida de datos (escritura)      Mostrar la salida de datos (**monitor**, impresora, archivo).

Escribir → **printf**.

```
printf("La suma es %i", suma);
```

- El **cartel** es "La suma es "
- %i que es una **máscara** de salida
- suma que es la **variable**

```
printf("El área es %i y el perímetro es %i", a, b);
```

}



## { Ejercicios

1. ¿Cuál será el resultado del siguiente programa?

```
main() {  
    int a=5, b;  
    b=a+2;  
    printf ("%d\n", b);  
    b=b+2;  
    printf ("%d\n", b);  
    b++;  
    printf ("%d\n", b);  
}
```

## { Ejercicios

2. Mostrar la salida resultante del siguiente segmento de programa.

```
char letra ='A';  
printf( " EST%c %cIEN ", letra, letra+1 );  
printf( "%c%c%c%c%c", letra+11, letra+4, letra+9, letra+14,letra+18 );
```

**La salida es:**

ESTA BIEN LEJOS

## { Ejercicios

3. Mostrar la salida que genera el siguiente programa:

**La salida que produce es:**

EL DÍA DOMINGO NO ES 18 SINO ES 19 DE MARZO

```
main()
{
    int valor = NUMERO;
    int numero=18;
    char cadena[]= DIA;

    printf("EL");
    printf(" DÍA %s",cadena);
    printf(" NO ES %d", numero);
    printf(" SINO ES %d", NUMERO);
    printf(" DE  %s","MARZO");

}
```

## { Ejercicios

4. Escribir un conjunto de instrucciones para que una variable se le asigne desde el teclado solo valores booleanos. Considerando que existen diferentes formas de representar valores de verdad (0,1,verdadero,falso,si,no,true,false).

```
char verdad ;  
printf("LECTURA DE FORMATOS DE VALORES BOOLEANOS( V,F, 1,0,S,N,T,F)"  
);  
scanf("%[10VFvfTFtfSNsn]",&verdad);  
printf("EL VALOR DE VERDAD INGRESADO ES %c\n",verdad);
```

}



## { Palabras reservadas

Son palabras proporcionadas por un lenguaje de programación para realizar determinadas operaciones, por lo que no pueden utilizarse como nombres identificadores (variables, constantes, funciones, etc.). Estas palabras reservadas pueden ser diferentes, dependerá del lenguaje de programación utilizado.

En C tenemos `main`, `break`, `while`, `for`, `switch`, `continue`, `printf`, `scanf`, `define`, etc.

**Atención:** Es importante saber que las palabras reservadas del código se escriben en minúsculas.

}

## { Comentarios en la codificación

El comentario puede utilizarse en cualquier parte del programa.

```
/* Esto es un comentario  
que ocupa varias líneas */
```

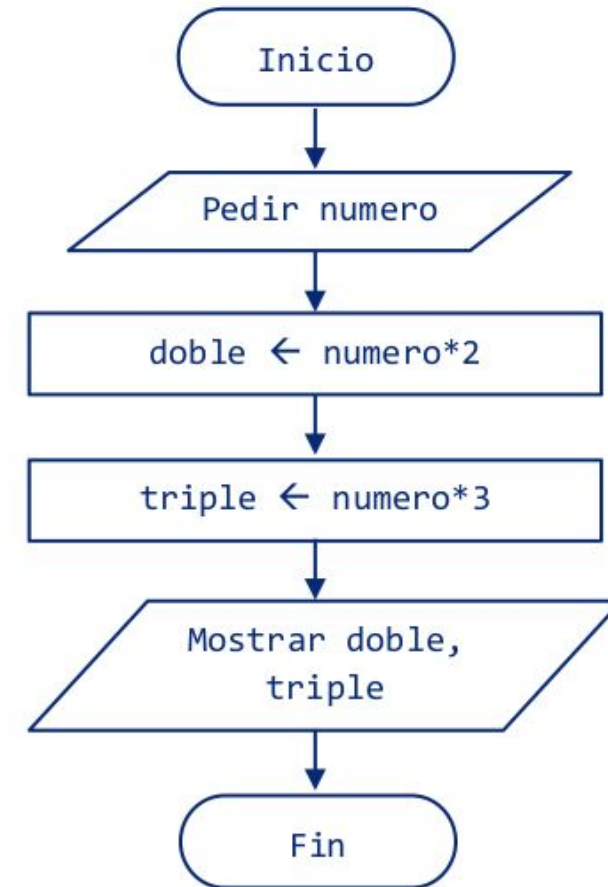
```
/* Ejemplo 3: Se ingresa el radio para mostrar  
el perímetro de la circunferencia */
```

```
#include <stdio.h>  
#define PI 3.1416  
main()  
{  
  
}
```

}

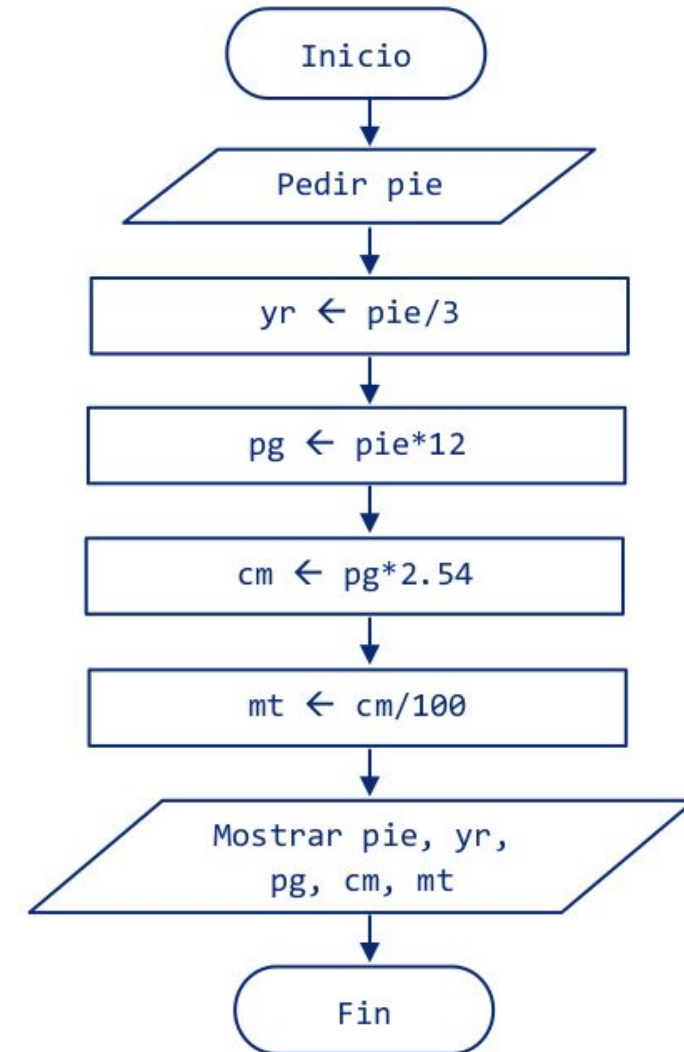
## { Ejercicio

Escribir un programa que lea un número entero y a continuación, visualice su doble y su triple.



## { Ejercicio

Escribir un programa para convertir una medida dada en pies a sus equivalentes en: a) yardas; b) pulgadas; c) centímetros; y d) metro. (1 pie: 12 pulgadas, 1 yarda= 3 pies, 1 pulgada= 2.54 cm, 1 metro= 100 cm). Leer el número de pies e imprimir el número de yardas, pies, pulgadas, centímetros y metros.







## Lista de ejercicios

- Realice un programa que tomado una cantidad expresada en metros lo transforme a su equivalente en kilómetros, centímetros y milímetros.
- Realice un programa que permita calcular la aceleración que tiene un cuerpo al conocer su velocidad inicial y final (m/s) en un instante de tiempo.
- Realice un programa que calcule la distancia de entre los puntos  $p1(x1,y1)$  y  $p2(x2,y2)$  considerando que  $d = ((X2-X1)^2 + (Y2-Y1)^2)^{1/2}$
- Realice un programa que descomponga un número ingresado por el usuario en su parte entera y decimal.



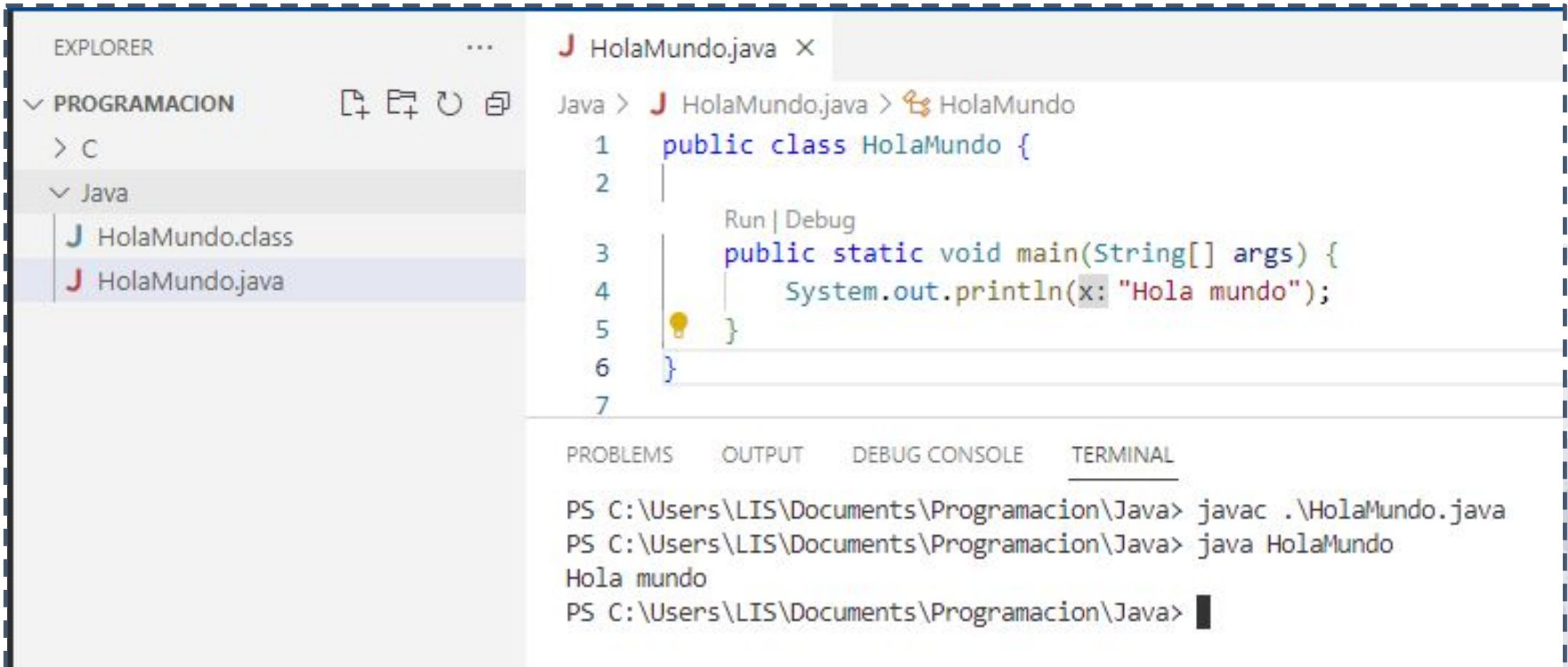


## Lista de ejercicios

- Escriba un programa que permita calcular la masa de aire con la siguiente fórmula:  $\text{masa} = (\text{presión} * \text{volumen}) / (0.37 * (\text{temperatura} + 460))$ . El ingreso de la masa, presión y volumen son cantidades enteras ingresadas por el usuario.
- En una concesionaria de vehículos se realizaron tres ventas de vehículos de alta gama a 3 clientes. Cada vehículo cuesta 30000, 29000 y 33000 usd. El gerente desea saber cuál es porcentaje (comisión) que cada vendedor se llevaría, lo que le pagará a cada uno de ellos (considerando el 4% por cada vendedor) y lo que le pagarán en conjunto (total).
- Un empleado de la empresa “Mi tienda es tuya” recibe un sueldo mensual, por su trabajo ha recibido un incremento del 25% sobre su sueldo anterior. Desarrolle un programa que permita calcular el nuevo sueldo del empleado.
- Un grupo de 8 amigos están preparando un viaje a la ciudad de Baños y desean saber el total que pagarán por esta aventura. Tres del grupo realizarán su viaje en taxi ejecutivo. Otros dos en una buseta turística y los tres restantes en bus que cobra 20% menos que el taxi.



Un programa Java empieza a ejecutarse por el método **main** de una clase seleccionada (**.java**).



The screenshot shows an IDE with the following components:

- EXPLORER:** A tree view on the left showing the project structure. It includes a folder named 'PROGRAMACION' which contains a sub-folder 'Java'. Inside 'Java', there are two files: 'HolaMundo.class' and 'HolaMundo.java'. 'HolaMundo.java' is currently selected.
- Editor:** The main area on the right displays the source code of 'HolaMundo.java'. The code is as follows:

```
1 public class HolaMundo {  
2  
3     Run | Debug  
4     public static void main(String[] args) {  
5         System.out.println(x: "Hola mundo");  
6     }  
7 }
```
- Terminal:** At the bottom of the IDE, the terminal output shows the commands used to compile and run the program:

```
PS C:\Users\LIS\Documents\Programacion\Java> javac .\HolaMundo.java  
PS C:\Users\LIS\Documents\Programacion\Java> java HolaMundo  
Hola mundo  
PS C:\Users\LIS\Documents\Programacion\Java> |
```

## Windows por terminal:

- Compilarlo:
  - `javac .\HolaMundo.java` ●
  - `javac HolaMundo.java`
- Ejecutar
  - `java HolaMundo`

## Linux (Ubuntu/Debian) por terminal:

- Compilarlo:
  - `javac HolaMundo.java`
- Ejecutar
  - `java HolaMundo`

El **compilador** traduce el código a **bytecode**, y lo guarda en un archivo **.class**.

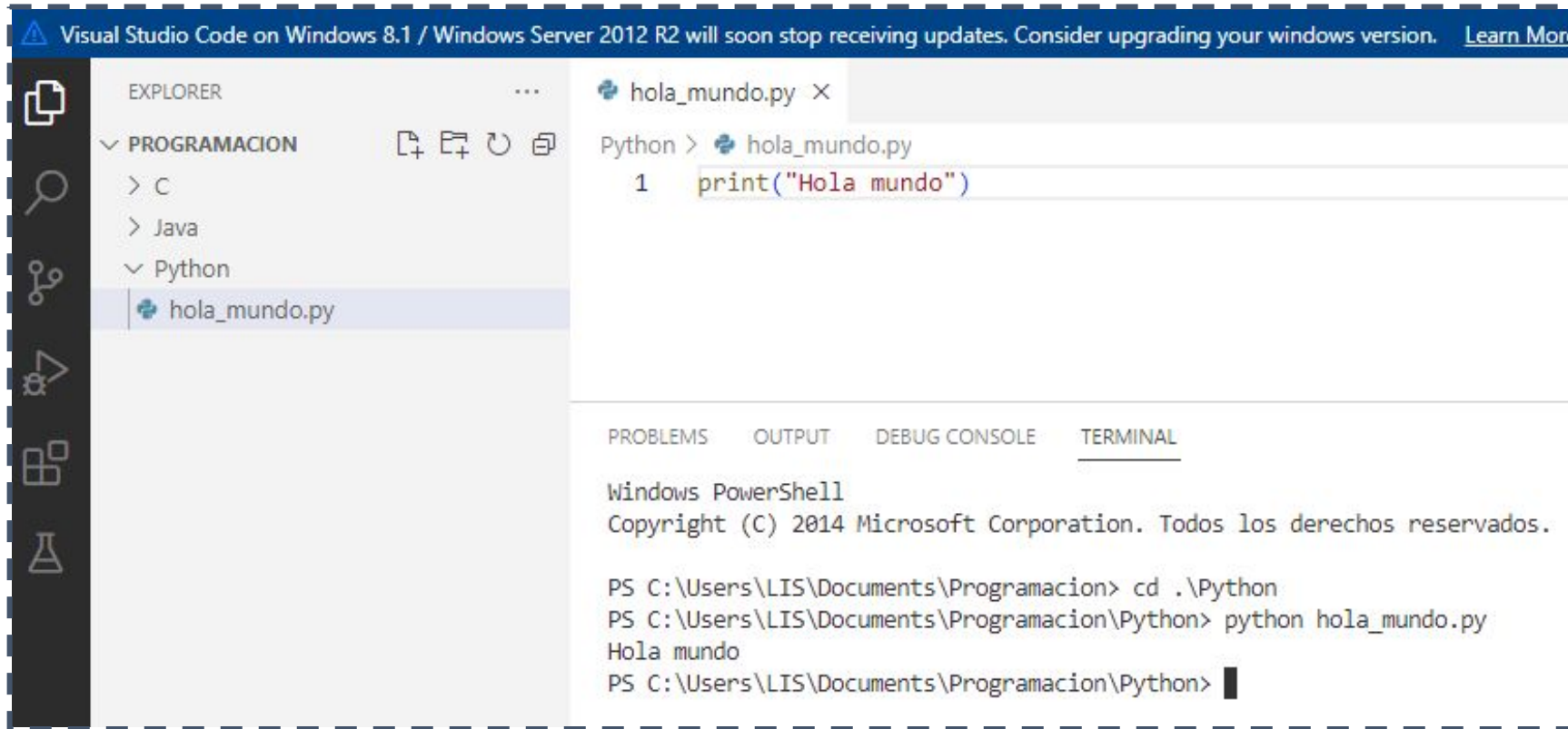
- Este no es código máquina nativo, sino un lenguaje intermedio que entiende la Máquina Virtual de Java (JVM).
- Para ejecutar, se usa la JVM (`java HolaMundo`). El programa no corre directo en el SO, sino dentro de la JVM, que interpreta/ejecuta el bytecode, traduciendo al lenguaje máquina en tiempo real.

## { Tipos de datos simples

- *Integer*: Sus valores son enteros
- *Long*: Sus valores son enteros más largos que los anteriores
- *Boolean*: Sus valores son true y false
- *Float*: Sus valores son números reales
- *Double*: Sus valores son números reales con mayor precisión
- *Character*: Sus valores son los caracteres disponibles como 'a'.
- *Void*: No tiene ningún valor
- *String*: Sus valores son secuencias de caracteres, como por ejemplo "Hola".

- `int`, `long`: sus valores son enteros.
- `float`, `double`: sus valores son números reales.
- `boolean`: sus valores son true o false.
- `char`: sus valores son de carácter.
- `void`: no tiene valores.





Visual Studio Code on Windows 8.1 / Windows Server 2012 R2 will soon stop receiving updates. Consider upgrading your windows version. [Learn More](#)

EXPLORER

- PROGRAMACION
  - C
  - Java
  - Python
    - hola\_mundo.py

hola\_mundo.py x

```
Python > hola_mundo.py
1 print("Hola mundo")
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

Windows PowerShell  
Copyright (C) 2014 Microsoft Corporation. Todos los derechos reservados.

```
PS C:\Users\LI5\Documents\Programacion> cd .\Python
PS C:\Users\LI5\Documents\Programacion\Python> python hola_mundo.py
Hola mundo
PS C:\Users\LI5\Documents\Programacion\Python> |
```

En Python el intérprete ejecuta el archivo línea por línea (**.py**), de arriba hacia abajo.



Universidad  
Nacional  
de Loja

# REFERENCIAS BIBLIOGRÁFICAS

- Goin, M. (2022). *Caminando junto al Lenguaje C*. Editorail UNRN Disponible en: [https://editorial.unrn.edu.ar/index.php/catalogo/346/view\\_bl/62/lecturas-de-catedra/26/caminando-junto-al-lenguaje-c?tab=get\\_mybooksTab&is\\_show\\_data=1](https://editorial.unrn.edu.ar/index.php/catalogo/346/view_bl/62/lecturas-de-catedra/26/caminando-junto-al-lenguaje-c?tab=get_mybooksTab&is_show_data=1)
- Guerra Salazar, J. E, Ramos Valencia, M. V, Vallejo Vallejo, G. E. (2023). Programando en C desde la práctica problemas resueltos. Puerto Madero Editorial. Disponible en: <https://dialnet.unirioja.es/servlet/libro?codigo=933288>
- Toro Bonilla, J. M. (2022). FUNDAMENTOS DE PROGRAMACIÓN: JAVA. Universidad de Sevilla, Editorial Universidad de Sevilla. Disponible en: <https://dialnet.unirioja.es/servlet/libro?codigo=871118>

An aerial photograph of the Universidad Nacional de Loja campus, showing various buildings, a large sports field, and surrounding greenery. A network graphic consisting of blue circles connected by dotted lines is overlaid on the right side of the image.

*Educamos para* **Transformar**

Redes UNL Oficiales



UNLoficial



UNLoficial



@UNLoficial



Universidad Nacional  
de Loja-UNL