



UNAH-CURLP
CENTRO UNIVERSITARIO REGIONAL
DEL LITORAL PACIFICO



TEAM HYDRA

Manual de Usuarios del Lenguaje de Programación **HYDRA**



Creado Por:

Jose Dulany Rueda Moreno

Ángel Antonio Pinto Paz

Hermes Josué Aguilera Flores



Introducción

Hydra es un lenguaje de programación interpretado, diseñado para combinar simplicidad, claridad de sintaxis y potencia en características modernas como la programación orientada a objetos, funciones, manejo de colecciones, estructuras de control y bibliotecas estándar.

Su propósito principal es servir como herramienta educativa y de experimentación, permitiendo a los programadores comprender conceptos fundamentales de la programación, desde operaciones básicas hasta arquitecturas más complejas con clases, herencia e interfaces.

Realizado en la clase de Diseño de Compiladores 2025 II PAC

“Y recuerda Hydra no busca ser el mejor si no

algo que te muestre la pasión que tuvimos en crearlo”



Contenido

Estructura básica de un programa en Hydra	4
Tipos de datos y variables	5
Operadores	6
Estructuras de control	8
Arreglos	10
Funciones	11
Programación Orientada a Objetos (POO)	13
Librerías e importaciones	14
Manejo de errores, excepciones, depuración y e.printStackTrace()	15
Desafío Hydra	17



Estructura básica de un programa en Hydra

```
# Haga un ejercicio para obtener datos de un empleado
start
    int edad;
    string nombre;
    float salario;
    bool activo;

    # Entradas
    output -> "=====ENTRADA DE DATOS=====";
    input -> nombre;
    input -> edad;
    input -> salario;
    input -> activo;

    # Salidas

    output -> "";
    output -> "Nombre" + nombre;
    output -> "Nombre" + edad;
    output -> "Nombre" + salario;
    output -> "Nombre" + activo;

    int nuevaEdad = edad + 1;
    output -> nombre + "Tendra" + nuevaEdad + "El siguiente año";

    if(activo){
        output -> nombre + "esta activo en la empresa";
    } else {
        output -> nombre + "No esta activo";
    }

end
```



Tipos de datos y variables

```
start

# ===== TIPOS DE DATOS BÁSICOS =====

# Entero
int edad = 25;

# Número decimal (float)
float altura = 1.75;

# Número decimal de doble precisión (double)
double precio = 199.99;

# Cadena de texto (string)
string nombre = "Laura";

# Carácter único
char inicial = 'L';

# Booleanos
bool esMayor = true;
bool tienePermiso = false;

# ===== VARIABLES CONSTANTES =====

const int AÑO_NACIMIENTO = 1997;
const string MENSAJE_BIENVENIDA = "Bienvenido a Hydra";
end
```



Operadores

```
start
```

```
# ===== OPERADORES ARITMÉTICOS =====
int a = 10;
int b = 3;

int suma = a + b;          # 13
int resta = a - b;         # 7
int multiplicacion = a * b; # 30
int division = a / b;      # 3 (entero)
int modulo = a % b;        # 1
int potencia = a ^ b;      # 1000 (a elevado a b)

# ===== OPERADORES DE ASIGNACIÓN =====
int x = 5;
x += 2;      # x = 7
x -= 1;      # x = 6
x *= 3;      # x = 18
x /= 2;      # x = 9
x %= 4;      # x = 1

# ===== OPERADORES RELACIONALES =====
bool mayor = a > b;        # true
bool menor = a < b;         # false
bool mayorIgual = a >= b;  # true
bool menorIgual = a <= b;  # false
bool igual = a == b;        # false
bool distinto = a != b;     # true
```



```
# ====== OPERADORES LÓGICOS ======
bool v1 = true;
bool v2 = false;

bool andLogico = v1 && v2; # false
bool orLogico = v1 || v2; # true
bool notLogico = !v1; # false

# ====== OPERADORES DE INCREMENTO Y DECREMENTO ======
int y = 5;
y++; # 6
y--; # 5

end
```



Estructuras de control

```
start

# ====== CONDICIONALES ======

int numero = 10;

# if
if (numero > 5) {
    output -> "El número es mayor que 5";
}

# if - else
if (numero % 2 == 0) {
    output -> "El número es par";
} else {
    output -> "El número es impar";
}

# if - else if - else
if (numero < 0) {
    output -> "Número negativo";
} else if (numero == 0) {
    output -> "Número cero";
} else {
    output -> "Número positivo";
}

# ====== SWITCH ======
int opcion = 2;

switch (opcion) {
    case 1:
        output -> "Opción 1 seleccionada";
    case 2:
        output -> "Opción 2 seleccionada";
    default:
        output -> "Opción no válida";
}
```



```
# ===== WHILE =====
int contador = 0;
while (contador < 3) {
    output -> "Contador: " + contador;
    contador++;
}

# ===== DO-WHILE =====
int i = 0;
do {
    output -> "Valor de i: " + i;
    i++;
} while (i < 3);

# ===== FOR CLÁSICO =====
for (int j = 0; j < 3; j++) {
    output -> "j = " + j;
}

# ===== FOR CON MÚLTIPLES VARIABLES =====
for (int k = 0, m = 5; k < 3; k++, m--) {
    output -> "k = " + k + ", m = " + m;
}

# ===== BREAK Y CONTINUE =====
for (int n = 0; n < 5; n++) {
    if (n == 2) {
        continue; # Salta el valor 2
    }
    if (n == 4) {
        break; # Termina el ciclo
    }
    output -> "n = " + n;
}

end
```



Arreglos

```
start
    # Arreglo vacío explícito
    int[] emptyIntArray = {};
    output -> "emptyIntArray.length = " + emptyIntArray.length;

    # Inicialización literal unidimensional
    int[] primes = {2, 3, 5, 7, 11};
    for (int i = 0; i < primes.length; i++) {
        output -> "primes[" + i + "] = " + primes[i];
    }

    # Modificar un elemento
    primes[2] = 13;
    output -> "primes[2] modificado = " + primes[2];

    # Arreglo bidimensional
    int[][] matrix = {
        {1, 2, 3},
        {4, 5, 6}
    };

    # Arreglo de cadenas
    string[] days = {"Lunes", "Martes", "Miércoles", "Jueves", "Viernes"};
    for (int k = 0; k < days.length; k++) {
        output -> "Día " + k + ": " + days[k];
    }

    # Arreglo más grande
    int[] big = {10, 20, 30, 40, 50, 60, 70, 80, 90, 100};
    for (int i = 0; i < big.length; i++) {
        output -> big[i];
    }
end
```



Funciones

```
start

# ===== FUNCIONES BÁSICAS =====

# Función sin parámetros y sin retorno
ft saludar() {
    output -> "Hola desde Hydra";
}

# Función con parámetros y sin retorno
ft mostrarNombre(string nombre) {
    output -> "Tu nombre es: " + nombre;
}

# Función con parámetros y retorno
ft sumar(int a, int b) -> int {
    return a + b;
}

# Función que retorna un string
ft obtenerSaludo(string nombre) -> string {
    return "Hola, " + nombre + "!";
}

# ===== FUNCIONES RECURSIVAS =====

ft factorial(int n) -> int {
    if (n == 0) return 1;
    else return n * factorial(n - 1);
}
```



```
# ===== FUNCIONES ESTÁTICAS EN CLASES =====

class Matematicas {
    static int cuadrado(int x) -> int {
        return x * x;
    }
}

# ===== LLAMADAS A FUNCIONES =====

saludar();
mostrarNombre("Laura");
int resultadoSuma = sumar(5, 7);
string mensaje = obtenerSaludo("Juan");
int fact5 = factorial(5);
int cuad4 = Matematicas.cuadrado(4);

end
```



Programación Orientada a Objetos (POO)

```
start

# ====== DEFINICIÓN DE CLASE BÁSICA ======


class Persona {
    string nombre;
    int edad;

    # Constructor
    Persona(string nombre, int edad) {
        self.nombre = nombre;
        self.edad = edad;
    }

    # Método para saludar
    ft saludar() -> string {
        return "Hola, soy " + self.nombre + " y tengo " + self.edad + " años.";
    }

    # Método para mostrar solo el nombre
    ft obtenerNombre() -> string {
        return self.nombre;
    }
}
```

```
# ====== CLASE CON MÉTODO ESTÁTICO ======


class Utilidades {
    static ft mensajeBienvenida() -> string {
        return "Bienvenido a Hydra";
    }
}

# ====== CREACIÓN Y USO DE OBJETOS ======


Persona p1 = new Persona("Laura", 28);
Persona p2 = new Persona("Juan", 35);

output -> p1.saludar();      # "Hola, soy Laura y tengo 28 años."
output -> p2.obtenerNombre(); # "Juan"

# Llamada a método estático sin crear objeto
output -> Utilidades.mensajeBienvenida();

end
```



Librerías e importaciones

```
# librerias.hy
# Demostración de imports y uso de funciones de la librería math

import math.sin;
import math.cos;
import math.tan;
import math.log;
import math.sqrt;
import math.pow;

start

    # Ángulo en radianes
    double angulo = 1.57079632679;    # ~ pi/2

    # Funciones trigonométricas
    double s = sin(angulo);
    double c = cos(angulo);
    double t = tan(angulo);

    # Logaritmo natural
    double ln10 = log(10.0);

    # Raíz cuadrada
    double raiz16 = sqrt(16.0);

    # Potencia (base, exponente)
    double dosALa8 = pow(2, 8);

    # (Opcional) Puedes imprimir si quieres ver el resultado,
    # pero no es necesario para "solo mostrarlo"
    # output -> s;
    # output -> c;
    # output -> t;
    # output -> ln10;
    # output -> raiz16;
    # output -> dosALa8;

end
```



Manejo de errores, excepciones, depuración y e.printStackTrace()

```
try {
    output -> "Ingrese primer operando:";
    input -> x;
    output -> "Ingrese segundo operando:";
    input -> y;
    resultado = div(x, y);
    output -> "Resultado: " + resultado;
} catch (string err) {
    output -> "¡Error!: " + err;
}
```

Manejo de TRY-CATCH

```
System.out.println("DEBUG PRIMARY: " + ctx.getText());
```

Manejo de Depuraciones en HydraExecutor

```
Ingrese el nombre del archivo (.hy): c.hy
DEBUG PRIMARY: main()
DEBUG PRIMARY: false
DEBUG PRIMARY: salir
DEBUG PRIMARY: "==== Calculadora Hydra ==="
==== Calculadora Hydra ===
DEBUG PRIMARY: "1) Sumar"
1) Sumar
DEBUG PRIMARY: "2) Restar"
2) Restar
DEBUG PRIMARY: "3) Multiplicar"
3) Multiplicar
DEBUG PRIMARY: "4) Dividir"
4) Dividir
DEBUG PRIMARY: "5) Salir"
5) Salir
DEBUG PRIMARY: "Seleccione opción:"
Seleccione opción:
Ingrese valor para op (int):
```

Vista del Manejo de Depuraciones en Programas



```
    } catch (Exception e) {
        System.err.println("Error de ejecución: " + e.getMessage());
        e.printStackTrace();
    }
```

e.printStackTrace(); en Main.java

```
java.lang.RuntimeException: Función no declarada: sin
    at HydraExecutor.visitPrimaryExpr(HydraExecutor.java:715)
    at HydraExecutor.visitPrimaryExpr(HydraExecutor.java:29)
    at HydraParser$PrimaryExprContext.accept(HydraParser.java:3559)
    at org.antlr.v4.runtime.tree.AbstractParseTreeVisitor.visit(AbstractParseTreeVisitor.java:18)
    at HydraExecutor.visitPostfixExpr(HydraExecutor.java:810)
    at HydraExecutor.visitPostfixExpr(HydraExecutor.java:29)
    at HydraParser$PostfixExprContext.accept(HydraParser.java:3432)
    at org.antlr.v4.runtime.tree.AbstractParseTreeVisitor.visit(AbstractParseTreeVisitor.java:18)
    at HydraExecutor.visitUnaryExpr(HydraExecutor.java:1685)
    at HydraExecutor.visitUnaryExpr(HydraExecutor.java:29)
    at HydraParser$UnaryExprContext.accept(HydraParser.java:3313)
    at org.antlr.v4.runtime.tree.AbstractParseTreeVisitor.visit(AbstractParseTreeVisitor.java:18)
    at HydraExecutor.visitExponentialExpr(HydraExecutor.java:1668)
    at HydraExecutor.visitExponentialExpr(HydraExecutor.java:29)
```

Vista del Programa con e.printStackTrace(); en Main.java

e.printStackTrace();

Si lo activas, Java imprime la ruta completa de qué métodos se llamaron y en qué línea exacta falló el programa, desde el inicio hasta el error.



Desafío Hydra

Entre más indagues en nuestro lenguaje, más errores, limitaciones y misterios encontrarás.

Pero eso es parte del viaje: Hydra está vivo, y tú puedes ayudarlo a crecer.

¿Tienes lo necesario para completar este lenguaje?

Cada error que encuentres es una oportunidad para aprender y aportar.

Primer parche desbloqueado:

Aquí tienes tu primera mejora lista para aplicar y comenzar a dejar tu huella.

Investiga, ajusta, rompe y vuelve a armar... hasta que Hydra sea más poderoso que nunca.

```
start
# Ejemplo de anidamiento infernal
    for (int i = 0; i < 10; i++) {
        if (i % 2 == 0) {
            while (true) {
                switch (i) {
                    case 2:
                        if (true) {
                            output -> "Nivel 5";
                        }
                        break;
                    default:
                        continue;
                }
            }
        } else {
            # ... más anidamiento
        }
    }
    int a = b = c = 5 + 4;

    int x = (((((((a + b) * c))))));
    const int MI_CONSTANTE = 10;
    # No es tan facil como piensas
end
```

Tienes lo necesario?



*Agradecimiento especial
Al Ph.D. Wilson Octavio Villanueva Castillo,
quien con su guía y dedicación hizo posible que alcanzáramos
nuestras metas.*

*Sin duda, una clase que nos dejó un valioso conocimiento y que, con
el paso del tiempo, aprendimos a valorar aún más.*

#TEAMHYDRA 