

H.U.L.K

Diego Manuel Viera Martínez

octubre , 2023



Proyecto II de programación

# 1. Expresiones H.U.L.K

## 1.1. Print:

Una de las instrucciones más simples en H.U.L.K (`print`), escribe en consola el valor de la expresión que pongas entre paréntesis después del `"print"`, en caso de no poner ninguna expresión pone una línea en blanco.

```
> print("Hello World !");  
Hello World !
```

## 1.2. Operaciones Aritméticas:

H.U.L.K cuenta con los operadores aritméticos: `+` (suma), `-` (resta), `*` (multiplicación), `/` (división), `%` (módulo) y potencia.

```
> print(((1 + 2) * 4) / 2);  
6
```

## 1.3. Funciones matemáticas básicas:

Contiene funciones matemáticas básicas para operaciones:

1. `'sqrt(<value>')` calcula la raíz cuadrada de un valor.
2. `'sin(<angle>')` calcula el seno de un ángulo en radianes.
3. `'cos(<angle>')` calcula el coseno de un ángulo en radianes.
4. `exp(<value>)` calcula el valor de 'e' elevado a un valor.
5. `'log(<base>, <value>')` calcula el logaritmo de un valor en una base dada.
6. `'rand()'` devuelve un número uniforme aleatorio entre 0 y 1 (ambos inclusive).

```
> print(sin(2 * PI) * 2 + cos(3 * PI / log(4, 64)));
```

## 1.4. Funciones:

Para declarar funciones hay que seguir una sintaxis:

```
function <nombre de la función>(parámetros) =><cuerpo de la función>
```

Ejemplo:

```
> function tan(x) =>sin(x) / cos(x);
```

## 1.5. Expresiones let-in:

En H.U.L.K es posible declarar variables usando la expresión let-in, que funciona de la siguiente forma:

```
> let x = PI/2 in print(tan(x));
```

En general, una expresión let-in consta de una o más declaraciones de variables, y un cuerpo, que puede ser cualquier expresión donde además se pueden utilizar las variables declaradas en el let. Fuera de una expresión let-in las variables dejan de existir.

El valor de retorno de una expresión let-in es el valor de retorno del cuerpo, por lo que es posible hacer:

```
> print(7 + (let x = 2 in x * x));
```

## 1.6. Condicionales

Las condiciones en HULK se implementan con la expresión if-else, que recibe una expresión booleana entre paréntesis, y dos expresiones para el cuerpo del if y el else respectivamente. Siempre deben incluirse ambas partes:

```
> let a = 42 in if (a% 2 == 0) print("Even") else print("odd");
```

Como if-else es una expresión, se puede usar dentro de otra expresión

```
> let a = 42 in print(if (a% 2 == 0) "even" else "odd");
```

### 1.6.1. Recursión

H.U.L.K soporta declaraciones de funciones recursivas:

```
> function fib(n) => if (n > 1) fib(n-1) + fib(n-2) else 1;
```

## 1.7. Errores

Presenta 3 tipos de errores:

1. Lexical Errors: Errores que se producen por la presencia de tokens inválidos.

```
> let 14a = 5 in print(14a);  
! LEXICAL ERROR: '14a' is not valid token.
```

2. Syntax Errors: Errores que se producen por expresiones mal formadas.

```
> let a = 5 in print(a);  
! SYNTAX ERROR: Missing closing parenthesis after 'a'.  
> let a = 5 inn print(a);  
! SYNTAX ERROR: Invalid token 'inn' in 'let-in' expression.  
> let a = in print(a);  
! SYNTAX ERROR: Missing expression in 'let-in' after variable 'a'.
```

3. Semantic Errors: Errores que se producen por el uso incorrecto de los tipos y argumentos.

```
> let a = "hello world" in print(a + 5);  
! SEMANTIC ERROR: Operator '+' cannot be used between 'string' and  
'number'.  
> print(fib("hello world"));  
! SEMANTIC ERROR: Function 'fib' receives 'number', not 'string'.  
> print(fib(4,3));  
! SEMANTIC ERROR: Function 'fib' receives 1 argument(s), but 2 were  
given.
```

## 1.8. ¿ Cómo iniciar y detener ?

Para inicializar H.U.L.K debes abrir su carpeta contenedora en consola y usar el comando(dotnet run) y listo puedes comenzar a usarlo. Para cerrar el programa escribe en el input (stop hulk) y regresará a la consola de su PC.