

Teoría en Computación

Semana 01: Introducción a la Teoría de la Computación

Mg. Roberto Zárate Mendoza



**Universidad
Tecnológica
del Perú**

Preguntas..... dudas ?

Dudas Consultas sobre la sesión anterior?

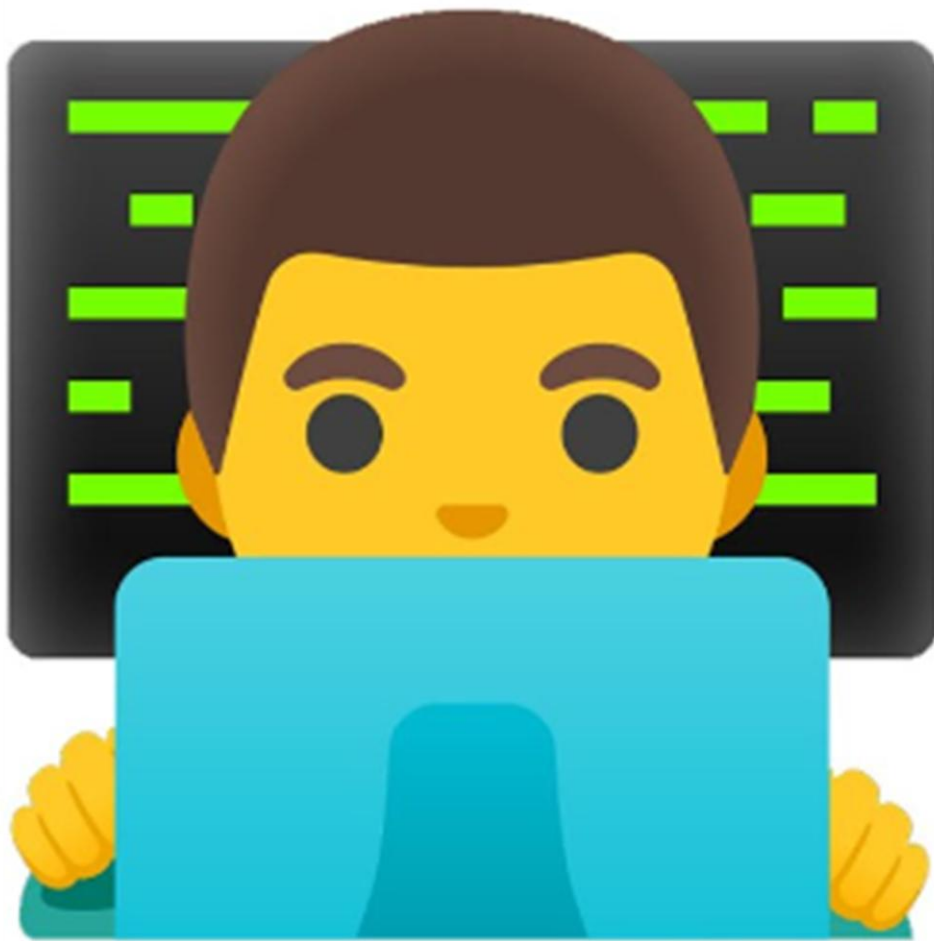
Saberes previos

Qué lenguajes de programación conocen?

Han compilado un programa en algún lenguaje de programación?

Agenda: Sesión

- **Pre-procesadores**
- **Ensambladores.**
- **Lenguajes de programación**



Logros de sesión

Al finalizar la sesión, el estudiante logra entender los conceptos de: pre-procesador y nociones básicas de lenguajes de programación.



Pre-procesadores

Un preprocesador es una herramienta o componente que se ejecuta antes de que el código fuente sea procesado por el compilador principal. Su función es realizar tareas preliminares para preparar el código fuente antes de que entre en las fases principales del compilador.

Funciones Principales

Inclusión de archivos : Incorpora archivos externos (como cabeceras o bibliotecas) en el código fuente. Por ejemplo, en C/C++, la directiva `#include` permite incluir archivos como `<stdio.h>`.

Expansión de macros : Reemplaza macros definidas con su valor correspondiente. Por ejemplo, en C/C++, `#define PI 3.1416` sustituye todas las ocurrencias de `PI` por `3.1416`.

Eliminación de comentarios : Elimina comentarios del código fuente para simplificar el trabajo del compilador.

Condicionales : Procesa directivas condicionales como `#ifdef`, `#ifndef`, `#if`, etc., para decidir qué partes del código deben ser incluidas o excluidas.

Optimización básica : Realiza pequeñas optimizaciones, como eliminar código redundante o inútil.

Pre-procesadores

Ejemplo 1:

En lenguajes como C o C++, el preprocesador es una etapa separada que transforma el código fuente antes de que el compilador lo analice. Por ejemplo:

```
#define MAX 100  
int array[MAX];
```

El preprocesador reemplaza MAX por 100 antes de que el compilador vea el código

Ejemplo 2:

El resultado sea 25 (el cuadrado de 5), pero obtiene un valor incorrecto. ¿Cuál es el problema y cómo se puede solucionar?

```
1  #define CUADRADO(x) x * x  
2  #include <iostream>  
3  |  
4  using namespace std;  
5  int main(){  
6      int resultado = CUADRADO(3 + 2);  
7      cout<< resultado;  
8      return 0;  
9  }
```

Pre-procesadores

Ejemplo 3:

Un programador define una macro para calcular el máximo de dos números:

El problema surge porque las macros en C/C++ realizan una **sustitución textual** antes de que el código sea compilado.

```
#define MAX(a, b) ((a) > (b) ? (a) : (b))
#include <iostream>

using namespace std;
int main(){
    int x = 5;
    int y = 10;
    int resultado = MAX(x++, y++);
    printf("Resultado: %d\n", resultado); // Resultado esperado: 10
    printf("x: %d\n", x);                // Valor final de x
    printf("y: %d\n", y);                // Valor final de y
    return 0;
}
```



Pre-procesadores

Ejemplo 4:

El comando `#define` se usa para definir variables con sus diferentes valores antes de la compilación

Los comandos `#ifndef`, `#ifdef` se ejecutan antes del proceso de compilación.

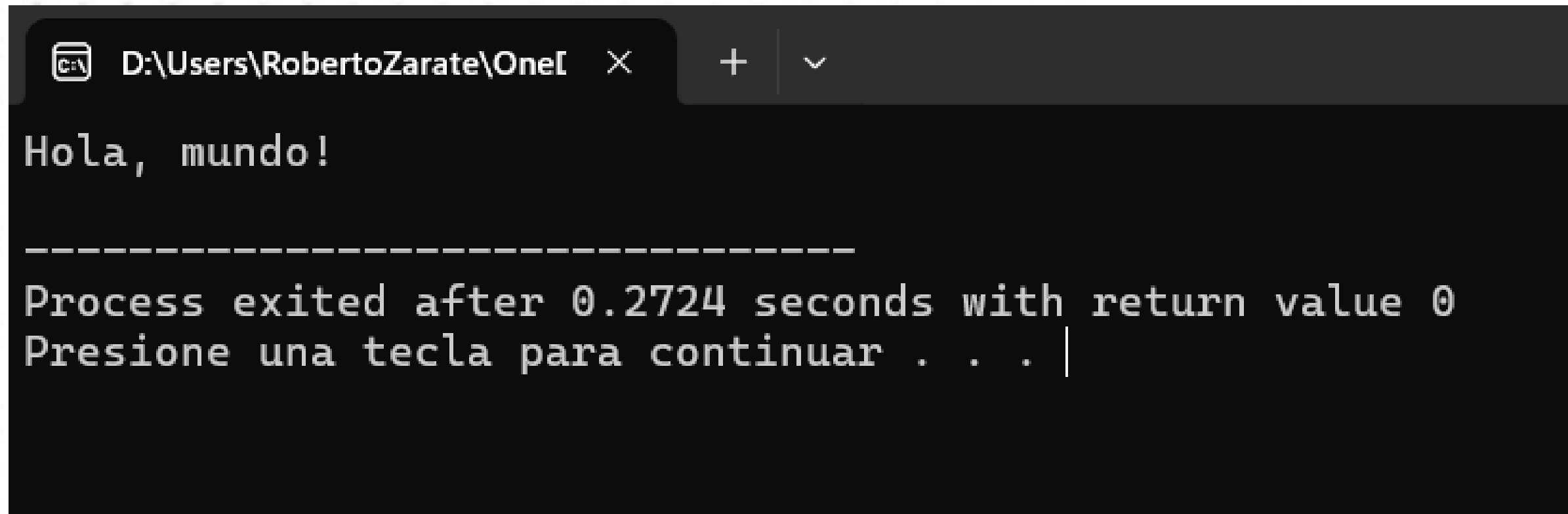
Directiva	Nombre completo	Comportamiento
<code>#ifdef X</code>	If Defined	Incluye el código si X está definido.
<code>#ifndef X</code>	If Not Defined	Incluye el código si X NO está definido.

```
1  #include <stdio.h>
2
3  // #define MODO_DEBUG // Define el macro MODO_DEBUG
4
5  int main() {
6      int x = 10;
7
8      #ifdef MODO_DEBUG
9          // Este bloque se compila SOLO si MODO_DEBUG está definido
10         printf("[DEBUG] Valor de x: %d\n", x);
11     #endif
12
13     printf("Hola, mundo!\n");
14     return 0;
15 }
```

Pre-procesadores

Ejemplo 4:

Resultado:



```
C:\> D:\Users\RobertoZarate\OneL  X + v
Hola, mundo!
-----
Process exited after 0.2724 seconds with return value 0
Presione una tecla para continuar . . . |
```

Ensambladores

Un ensamblador es una herramienta que convierte código escrito en lenguaje ensamblador (un lenguaje de bajo nivel cercano al hardware) en código máquina ejecutable por el procesador.

```
section .data
    mensaje db "Hola, mundo!", 0

section .text
    global _start

_start:
    mov eax, 4      ; syscall: write
    mov ebx, 1      ; file descriptor:
                    ; stdout
    mov ecx, mensaje ; dirección del
mensaje
    mov edx, 12     ; longitud del
mensaje
    int 0x80        ; interrupción del
sistema

    mov eax, 1      ; syscall: exit
    xor ebx, ebx    ; código de salida: 0
    int 0x80
```

Lenguajes de Programación

Los lenguajes de programación cuentan con los elementos antes mencionados.

Definición: Un lenguaje de programación es un conjunto de reglas y sintaxis que permiten a los programadores comunicarse con las computadoras para realizar tareas específicas.

Historia: Breve recorrido histórico desde los primeros lenguajes (como Fortran y COBOL) hasta los lenguajes modernos (como Python y JavaScript).

Clasificación de los Lenguajes de Programación

- **Lenguajes de bajo nivel:** Ensamblador, lenguaje máquina.
- **Lenguajes de alto nivel:** C, C++, Java, Python.
- **Lenguajes de muy alto nivel:** MATLAB, R.
- **Lenguajes de propósito específico:** SQL, HTML, CSS.

Lenguajes de Programación

Paradigmas de Programación:

- ❑ **Procedural** (C, Pascal): Orientado a procedimientos
- ❑ **Orientado a objetos** (Java, C++): Define Clases, objetos
- ❑ **Funcional** (Haskell, Lisp).
- ❑ **Lógico** (Prolog): Es declarativo, se centra en que se debe resolver y no en cómo se debe resolver, no secuencial
- ❑ **Scripting** (Python, JavaScript): No requieren compilación previa, Tipado dinámico (variables no se declaran con tipo) y estructuras flexibles, Permiten escribir código rápido y legible.

¿Todo claro?



Preguntas de Cierre:

- ¿Qué es un pre-procesador?
- ¿Cuáles son los lenguajes de programación:
 - de alto nivel?
 - de bajo nivel?





**Universidad
Tecnológica
del Perú**