



Día, Fecha:	16 de agosto del 2024
Hora de inicio:	19:00 a 20:40

0781 ORGANIZACION DE LENGUAJES Y COMPILEDORES 2

Daniel Enrique Santos Godoy



SEGUNDO SEMESTRE - 2023

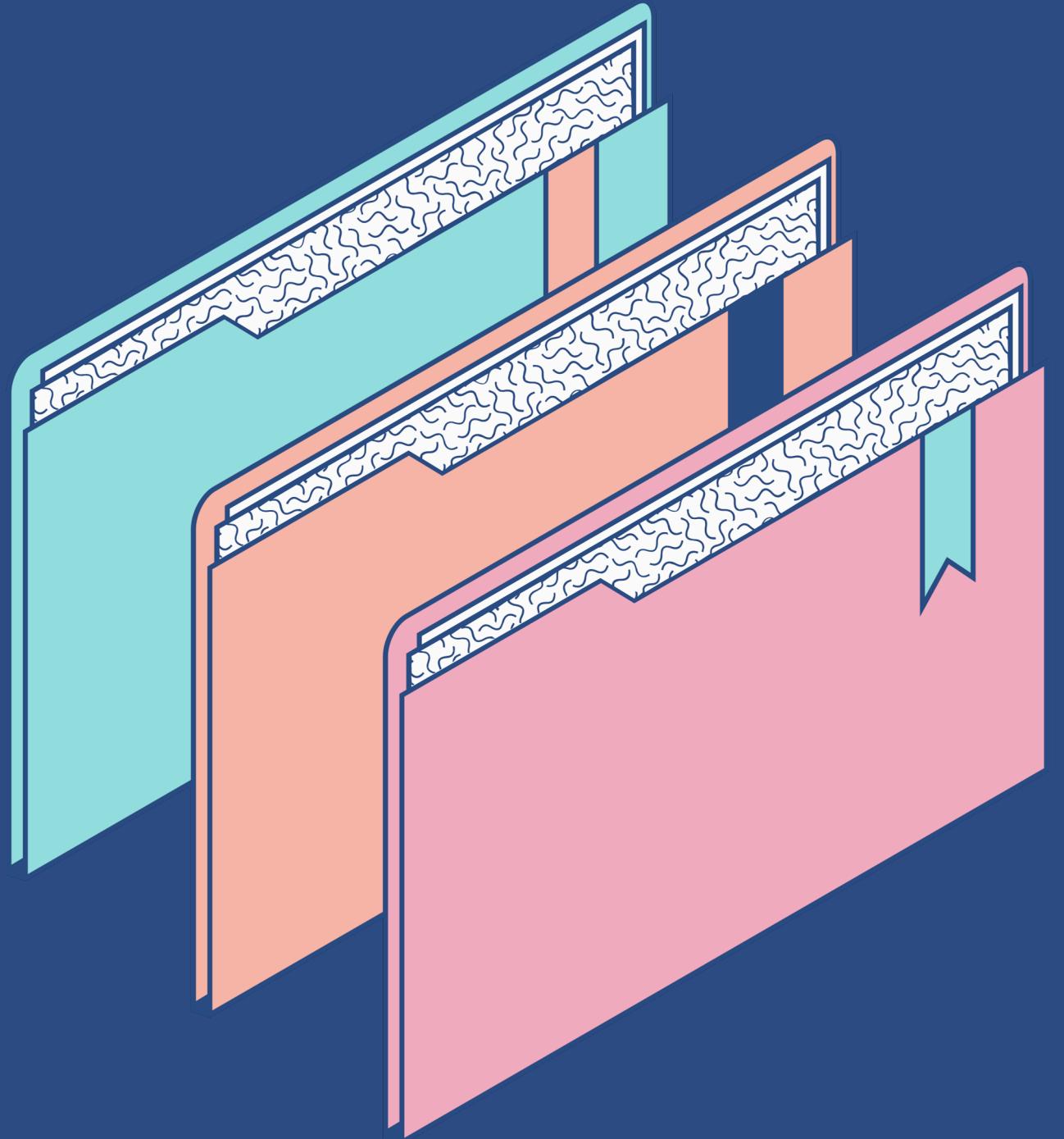
Organización de Lenguajes y Compiladores 2

Clase 5 - Arreglos y accesos

Agenda

TEMAS CLAVE QUE SE
DEBATIRÁN EN ESTA
PRESENTACIÓN

- ¿Dudas?
- Enunciado del Proyecto
- Expresiones
- Instrucciones
- Errores en Peggy
- Ejemplo Práctico





¿Qué es una expresión?



Una expresión es una combinación de valores, variables, operadores y llamadas a funciones que, al ser evaluada, produce un valor. Es como una pequeña fórmula matemática que el programador utiliza para obtener un resultado.

$2 + 3$

$x * y$

suma(2, 4);

Interpreter



¿Qué es una instrucción?



Una instrucción, por otro lado, es una orden que le damos al computador para que realice una acción específica. No necesariamente produce un valor, sino que ejecuta una tarea.

- `print("Hola, mundo!")`
- `if (x > 5): ...`
- `for i in range(10): ...`

Interpreter

Errores

La aplicación debe ser capaz de reportar los diversos errores al momento de validar el código. Los tipos de errores que deberán ser identificados en los reportes son:

- Léxicos
- Sintácticos

Interpreter

Errores

La herramienta Peggy trata los errores tanto léxicos como sintácticos de una sola manera, ambos los toma como producciones no esperadas.

```
▼ {StartRules: Array(1), SyntaxError: f, parse: f} ⓘ
  ► StartRules: ["Start"]
  ▼ SyntaxError: f peg$SyntaxError(message, expected, found, location)
    ► buildMessage: f (expected, found)
      length: 4
      name: "peg$SyntaxError"
    ► prototype: Error (format: f)
      arguments: (...)

      caller: (...)

      [[FunctionLocation]]: parser.js:13
    ► [[Prototype]]: f ()
    ► [[Scopes]]: Scopes[3]
  ► parse: f peg$parse(input, options)
  ► [[Prototype]]: Object
```

Interpreter

Errores

Por lo tanto para poder obtener los diferentes tipos de errores necesitamos generar funciones de diferenciación.

```
83 function isLexicalError(e) {  
84     const validIdentifier = /^[a-zA-Z$_][a-zA-Z0-9$_]*$/;  
85     const validInteger = /^[0-9]+$/;  
86     const validRegister = /^[a-zA-Z][0-9]+$/;  
87     const validCharacter = /^[a-zA-Z0-9$_\\[\\]]$/;  
88  
89     if (e.found) { ...  
90     }  
91  
92     return false; // Error sintáctico  
93 }
```

Interpreter

Ejemplo Práctico



SEGUNDO SEMESTRE - 2023

Gracias por su
atención..



Clase 5