Table of Contents

Preface

- The Art and Science of Parsing
- Why Parsing Expression Grammars?
- About This Documentation

Chapter 1: Understanding PEG Parsers

- 1.1 Introduction to Parsing Expression Grammars
- 1.2 PEG vs. Context-Free Grammars
- 1.3 PEG Operators and Notation
- 1.4 Recursive Descent Parsing
- 1.5 Packrat Parsing and Memoization
- 1.6 TinyPEG Implementation Preview

Chapter 2: TinyPEG Library Overview

- 2.1 Architecture and Design Philosophy
- 2.2 Core Components
- 2.2.1 Expression Class
- 2.2.2 Reference Class
- 2.2.3 ParserContext Class
- 2.2.4 ParseError Class
- 2.3 Parser Components
- 2.3.1 PEGParser Class
- 2.3.2 Rule Class
- 2.4 Syntax Tree Components
- 2.4.1 GrammarNode Class
- 2.4.2 DebugVisitor Class
- 2.5 Testing the Library

Chapter 3: Building Your First Parser

- 3.1 Setting Up the Environment
- 3.2 Creating a Simple Numeric Expression Parser
- 3.3 Adding Support for Operators
- 3.4 Handling Parentheses and Precedence
- 3.5 Building an Abstract Syntax Tree
- 3.6 Evaluating Expressions
- Summary

Chapter 4: Example Parsers and Applications

- 4.1 Calculator Examples
- 4.1.1 Simple Calculator
- 4.1.2 Advanced Calculator
- 4.1.3 Calculator Base Classes
- 4.1.4 Testing and Usage
- 4.2 Language Parser Examples
- 4.2.1 Basic Language Constructs
- 4.2.2 TinyCL Language Variants
- 4.3 Complete TinyCL Implementation
- 4.3.1 TinyCL Features
- 4.3.2 Comprehensive Test
- 4.3.3 Multi-Target Compilation
- 4.4 Running the Examples
- 4.5 Example Organization
- Summary

Chapter 5: Creating TinyCL - A Complete Programming Language

- 5.1 TinyCL Language Overview
- 5.1.1 Language Features
- 5.1.2 Complete Grammar Specification
- 5.2 Complete Implementation Overview
- 5.2.1 Parser Implementation
- 5.3 Building the Abstract Syntax Tree

• 5.3.1 AST Node Classes
• 5.3.2 Tree Construction
• 5.4 Semantic Analysis
• 5.4.1 Symbol Table
• 5.4.2 Type Checking
• 5.5 Interpreter Implementation
• 5.5.1 Runtime Environment
• 5.5.2 Expression Evaluation
• 5.5.3 Statement Execution
• 5.6 Example Programs and Testing
• Summary

Appendix A: TinyPEG Library Reference

• A.1 TinyPEG Implementation Overview
• A.1.1 Core Architecture
• A.1.2 Grammar Definition
• A.1.3 TinyPEG Expression Classes
• A.1.4 Complete Example
• A.1.5 TinyPEG Semantics
• A.2 Common TinyPEG Patterns
• A.2.1 Whitespace Handling
• A.2.2 Identifiers
• A.2.3 Numbers
• A.2.4 Strings
• A.2.5 Comments
• A.2.6 Expressions with Precedence
• A.3 Comparison with Regular Expressions
• A.4 TinyPEG Implementation Details
• A.4.1 Memoization
• A.4.2 Left Recursion Handling
• A.4.3 Error Reporting
• A.4.4 AST Building
• A.4.5 Whitespace Handling
• A.5 TinyPEG vs Other PEG Libraries
• A.6 Complete API Reference
• Summary

Appendix B: Testing Framework

- B.1 Unit Testing Parsers
- B.2 Test Case Design
- B.2.1 Valid Input Tests
- B.2.2 Invalid Input Tests
- B.2.3 Performance Tests
- B.2.4 Regression Tests
- B.3 Debugging Techniques
- B.3.1 Tracing
- B.3.2 Visualization
- B.3.3 Step-by-Step Execution
- B.3.4 Simplified Test Cases
- B.3.5 Logging
- B.4 Testing Tools
- B.5 Continuous Integration
- Summary

Appendix C: TinyCL Language Reference

- C.1 Language Overview
- C.2 Complete Syntax Reference
- C.2.1 Program Structure
- C.2.2 Statements
- C.2.3 Complete Expression System
- C.2.4 Data Types and Literals
- C.2.5 Identifiers
- C.3 Standard Library
- C.4 Example Programs
- C.4.1 Hello, World!
- C.4.2 Factorial with Functions
- C.4.3 Fibonacci Sequence
- C.4.4 Array Processing
- C.5 Language Features Summary
- C.5.1 Implemented Features
- C.5.2 Current Limitations
- C.5.3 Possible Future Extensions

- Summary