苏州大学实验报告

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 院系 | 计算机学院 | | 年级专业 | | 21计科 | | 姓名 | 方浩楠 | 学号 | 2127405048 |
| 课程名称 | | 编译原理课程实践 | | | | | | | 成绩 |  |
| 指导教师 | | 王中卿 | | 同组实验者 | | 无 | | 实验日期 | 2023.11.23 | |

|  |  |
| --- | --- |
| 实 验 名 称 | 实验 |

1. 实验目的

掌握基于PLY的解析技术： 通过使用Python Lex-Yacc（PLY），深入理解编译原理中的语法分析过程。

理解并实现简单的解析器： 学习如何构建一个能够处理基本Python语句（赋值、四则运算、print语句）的解析器。

语法树的构建与理解： 学习如何从解析过程中构建语法树，并理解其结构与用途。

实现语法制导翻译： 理解并实践如何通过语法树进行语法制导翻译，包括变量值的存储和运算结果的计算。

1. 实验内容

使用PLY进行解析：

利用PLY完成对指定Python程序（example2.py）的语法分析。

解析内容包括赋值语句、四则运算和print语句。

构建无二义性的语法规则：

根据提供的四则运算语法规则，构建一个无二义性的解析规则。

生成语法树：

解析过程中构建语法树，以展示解析结果的结构。

实现语法制导翻译：

在语法树节点中添加属性value以保存节点值。

创建变量表用于存储每个变量的值。

通过深度优先遍历实现整个语法树的语义分析与计算。

1. 实验步骤和结果

项目结构图:

**experiment10/**

**|**

**|-- data/**

**| |-- 0.py**

**| |-- example.py**

**|**

**|-- docs/**

**| |-- 实验报告.docx**

**|**

**|-- lexer/**

**| |-- \_\_init\_\_.py**

**| |-- py\_lex.py # 词法分析器文件，定义了解释器如何将输入文本分解成一系列标记。**

**|**

**|-- nodes/**

**| |-- \_\_init\_\_.py**

**| |-- node.py # 定义了节点类，用于构建抽象语法树（AST），每个节点代表源代码中的一个构造。**

**|**

**|-- parser/**

**| |-- \_\_init\_\_.py**

**| |-- parsetab.py # Bison/Flex工具生成的文件，包含了解析表，由py\_yacc.py使用。**

**| |-- py\_yacc.py # 语法分析器文件，包含了解释器如何根据词法标记构建AST的规则。**

**|**

**|-- utils/**

**| |-- \_\_init\_\_.py**

**| |-- data\_translator.py # 数据转换器，可能包含了执行AST节点和执行语义动作的逻辑。**

**| |-- text\_utils.py # 文本处理工具，提供文本处理相关的辅助函数。**

**|**

**|-- venv/ # 包含Python虚拟环境的相关文件，用于隔离项目依赖，确保不同项目间的依赖不会相互冲突。**

**|**

**|-- main.py # 主执行文件，包含了启动解释器的入口代码，可能会处理命令行参数、读取文件输入等。**

**|-- readme.md # 项目的README文件，通常包含项目的概述、安装指南、使用方法和其他重要信息。**



实验步骤

步骤 1: 环境准备

创建并激活Python虚拟环境。

安装必要的库，比如PLY。

步骤 2: 词法分析器的构建（lexer/py\_lex.py）

定义语言的词汇规则，如数字、运算符等。

使用PLY的lex工具生成词法分析器。

步骤 3: 节点定义（nodes/node.py）

实现Node类，用于创建AST的节点。

定义节点的数据结构，包括数据、子节点等属性。

步骤 4: 语法分析器的构建（parser/py\_yacc.py）

定义语言的语法规则，如表达式、赋值语句等。

使用PLY的yacc工具生成语法分析器。

步骤 5: 数据转换与执行（utils/data\_translator.py）

实现将AST节点转换为可执行代码的逻辑。

实现语义动作，如变量赋值、表达式求值等。

步骤 6: 文本工具的实现（utils/text\_utils.py）

编写辅助函数处理文本，如清洗输入文本等。

步骤 7: 主程序执行（main.py）

综合lexer、parser和utils模块，实现解释器的完整功能。

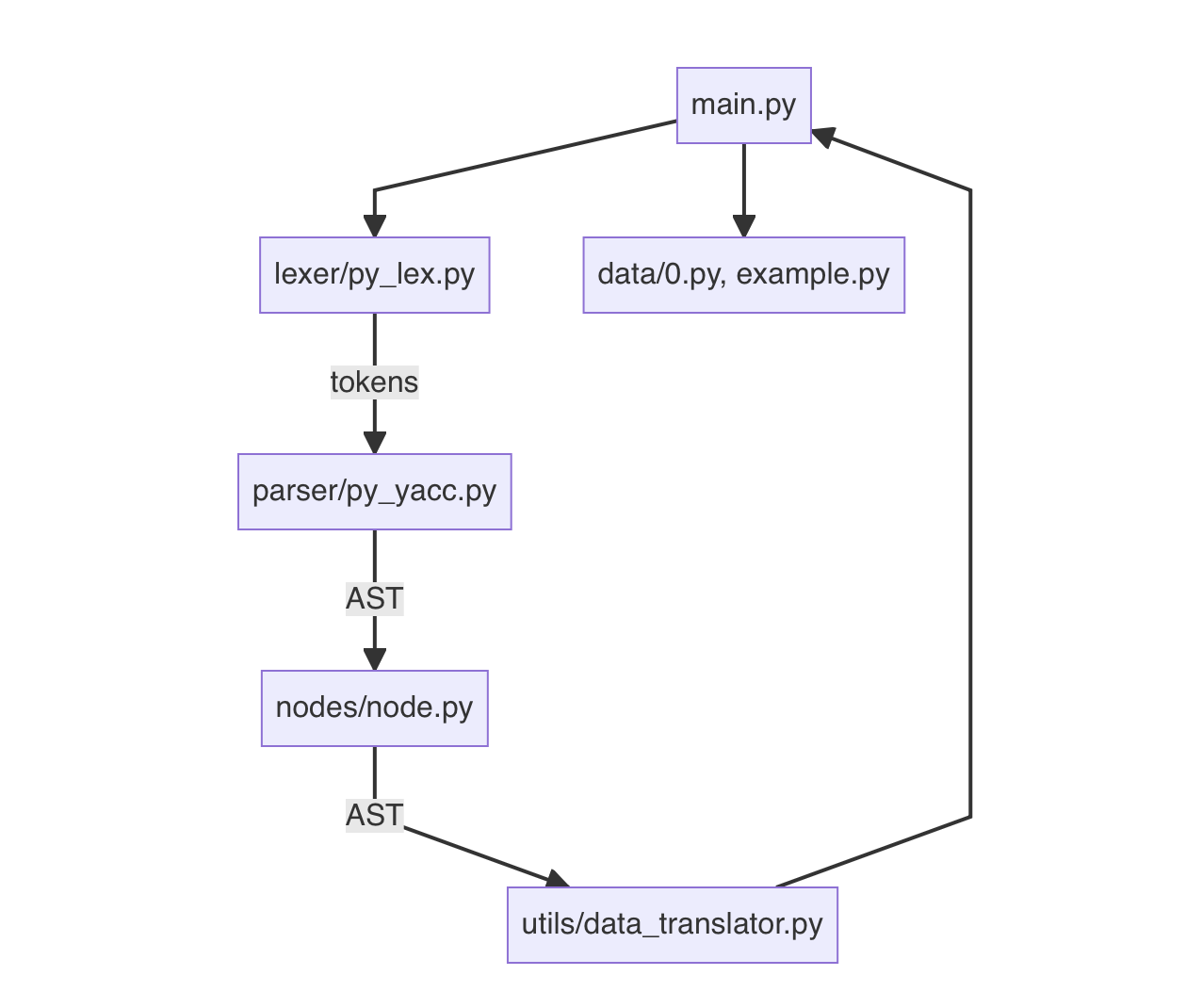
读取输入文件，进行词法分析、语法分析、AST构建和语义执行。

步骤 8: 测试和验证

使用data/0.py和data/example.py进行测试。

验证解释器的输出是否符合预期。

下面的是该实验的流程图



example.py内容:

**a = 1**

**b = 2**

**c = a + b**

**d = c - 1 + a**

**print(c)**

**print(a, b, c)**

输出结果:

**/Users/fanghaonan/File/study/Soochow\_University/Compilers\_Principles/experiment10/venv/bin/python /Users/fanghaonan/File/study/Soochow\_University/Compilers\_Principles/experiment10/main.py**

**+ [PROGRAM]**

**+ [STATEMENTS]**

**+ [STATEMENTS]**

**+ [STATEMENTS]**

**+ [STATEMENTS]**

**+ [STATEMENTS]**

**+ [STATEMENTS]**

**+ ['STATEMENT']**

**+ [ASSIGNMENT]**

**+ a**

**+ =**

**+ 1**

**+ ['STATEMENT']**

**+ [ASSIGNMENT]**

**+ b**

**+ =**

**+ 2**

**+ ['STATEMENT']**

**+ [OPERATION]**

**+ c**

**+ =**

**+ a**

**+ +**

**+ b**

**+ ['STATEMENT']**

**+ [OPERATION]**

**+ d**

**+ =**

**+ c**

**+ -**

**+ 1**

**+ +**

**+ a**

**+ ['STATEMENT']**

**+ [PRINT]**

**+ c**

**+ ['STATEMENT']**

**+ [PRINT]**

**+ a**

**+ b**

**+ c**

**{'a': 1.0, 'b': 2.0, 'c': 3.0, 'd': 3}**

**进程已结束，退出代码为 0**

**文本

描述已自动生成**

**图形用户界面, 文本

描述已自动生成**

可以看到,最终a,b,c,d四个变量的值被成功计算,并且保存在了v\_table = {}中

data\_translator.py的主要结构:

**def update\_v\_table(self, name, value):**

**"""更新变量表中的变量值。"""**

**self.v\_table[name] = value**

**def translate(self, node):**

**"""递归地转换或执行给定AST节点，并返回结果。"""**

**# 递归遍历子节点**

**for c in node.get\_children():**

**self.translate(c)**

**# 处理赋值节点**

**if node.get\_data() == '[ASSIGNMENT]':**

**# 相关逻辑...**

**# 处理操作节点**

**elif node.get\_data() == '[OPERATION]':**

**# 相关逻辑...**

**# 处理打印节点**

**elif node.get\_data() == '[PRINT]':**

**# 相关逻辑...**

1. 实验总结

理论与实践的结合：

本实验强化了编译原理理论知识与实践技能的结合，特别是在语法分析和语法制导翻译方面。

问题解决与思维拓展：

在解决解析过程中遇到的问题时，增强了问题解决能力和逻辑思维。

对编译原理的深入理解：

通过实践加深了对编译原理中诸如词法分析、语法分析、语法树构建和语义分析的理解。

反思与未来展望：

反思实验过程中的不足，规划未来在编译原理及相关领域的学习路径。