華南師範大學

本科学生实验（实践）报告

院　　系：计算机学院

实验课程： 编译原理

实验项目： Rust 语言词法分析

指导老师： 黄煜廉

开课时间： 2024 ~ 2025 **年度第** 1 **学期**

专　　业： 网络工程

班　　级： 2 班

学生姓名： 肖翔

华南师范大学教务处

华南师范大学实验报告

学生姓名 肖翔　　学　　号 20222132002

专　　业 网络工程　　年级、班级 2022 级 网络工程二班

课程名称 编译原理　　实验项目 Rust 分词器

实验类型　□验证 □设计 □综合　实验时间 2024 年 9 月 4 日

实验指导老师　　实验评分

一、实验内容

1. 把 Rust 源代码中的各类单词（记号）进行拼装分类。

Rust 语言包含了几种类型的单词（记号）：标识符，关键字，字面量数（包括二进制、十进制、八进制和十六机制的整数、浮点数），字面量字符串、注释、分隔符和运算符号等。

2. 要求应用程序应为 Windows 界面（窗口式图形界面）：使用对话框打开一个 Rust 源文件，并使用对话框列出所有可以拼装的单词（记号）及其分类。

3. 实验设计实现需要采用的编程语言：C++程序设计语言

4. 根据实验的要求组织必要的测试数据（要能对实验中要求的各类单词都能做完备的测试）。

二、实验目的

通过实现一个 Rust 语言的分词器，深入理解编译原理中的词法分析阶段，包括如何处理不同类型的词法单元。通过设计一个能够处理多行注释和字符串字面量的分词器，学习状态机的原理和应用，特别是在处理复杂语言结构时的状态转换。

三、实验文档：

（一）系统概述

1. 系统结构

枚举定义（TokenID）：定义了所有可能的词法单元的类型。

映射定义：使用 map 定义了几个映射，用于将字符串映射到词法单元 ID，例如关键词、宏、转义字符等。

数据结构定义：定义了一个结构体 TokenStru 来存储分词的结果。

变量定义：定义了一些全局变量，如用于读取源代码的缓冲区 buffer，当前处理位置 pos，以及用于标记注释状态的标志 flag1、flag2 等。

辅助函数：hexCharToInt 函数用于将十六进制字符转换为数值。

核心分词函数：GetToken 函数是分词的核心，它根据当前字符和上下文决定下一个词法单元的类型。

初始化函数：init 函数用于初始化关键词、宏、转义字符等映射。

主函数：main 函数负责读取输入文件，调用分词函数，并输出分词结果。

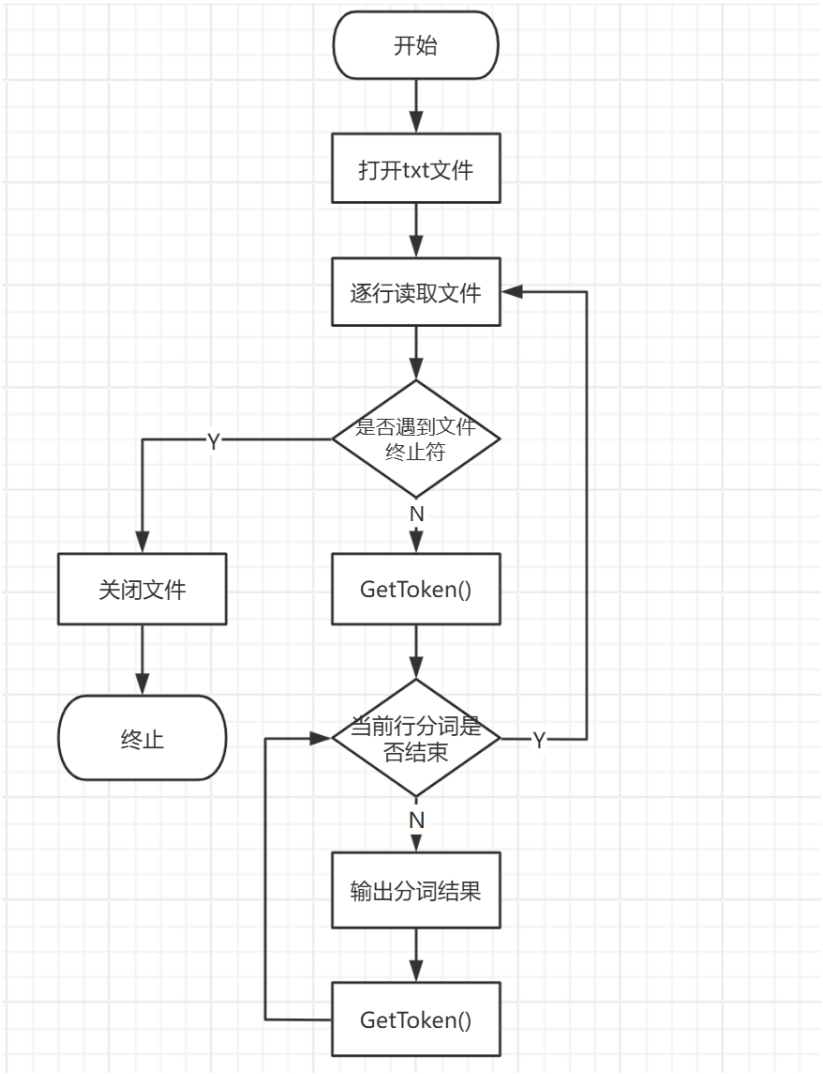
2. 数据结构的选择

枚举（Enum）：用于定义词法单元的类型，清晰且易于管理。

映射（Map）：用于快速查找字符串对应的词法单元类型，效率高。

结构体（Struct）：TokenStru 结构体用于存储分词结果，包括类型、数值、字符串等信息。

3. 程序整体大致流程的流程图如下。



（二）GetToken 函数执行的大致流程及其设计与实现

作为分词器的核心，GetToken 函数设计时主要采用了分支结构（if-elseif-else， switch-case）。通过当前词语的上下文（或者更准确地说根据当前词语的某些字符）判断其词法单元类型。GetToken 函数使用 pos 索引扫描词语。以下是 GetToken 函数执行的大致流程：

1. 跳过空白字符：函数首先跳过空格和制表符（buffer[pos] == 32 || buffer[pos] == 9），除非当前处于注释状态。

2.处理注释：如果当前处于多行注释（flag1）、单行注释（flag2）或文件注释（flag3）状态，函数会读取注释内容直到注释结束。这样的处理方式非常有利于对多行注释的标记。

3.处理字符串字面量：如果当前字符是双引号（"）或者已经处于字符

串字面量状态（flagStr），函数会读取字符串直到遇到下一个双引号。flagStr 为辅助标记跨行字符串而存在。

4.处理数值字面量：如果当前字符是数字，函数会根据数字的前缀（如 0b 表示二进制，0o 表示八进制，0x 表示十六进制）来确定数值类型，并通过进制计算字符对应的十进制数值。

5.处理字节字符串和原始字符串：如果当前字符序列是字节字符串（b"）或原始字符串（r#"），函数会读取相应的字符串类型。

6.处理字符字面量：如果当前字符是单引号（'）并且其后只有一个字符被另一个单引号包围，函数会读取字符字面量。

7.处理标识符和关键字：如果当前字符是字母或下划线，函数会读取标识符，并通过映射检查它是否是关键字、宏或类型名。

8.处理生命周期：如果当前字符是单引号并且后续字符是字母（字母未被另一个单引号包围，字母后可跟数字），函数会读取生命周期。

9.处理操作符和分隔符：函数通过 switch-case 检查当前字符是否是操作符或分隔符，并相应地设置词法单元类型。

10.处理转义字符：通过映射实现。

11.若 pos 已扫描至 buffer 字符数组的末尾（'\0'），说明当前行的词语已扫描完毕，函数会将 token 标记为 ENDINPUT，令程序继续读取文件的下一行。

（三）关键算法设计思路

在 GetToken 函数中，多行注释和字符串的跨行处理是关键的。其他词语并不涉及跨行，它们只需在一行内就能被识别，因此处理方法相较统一和简单。而多行注释和字符串由于存在多行，它们的内容并不都与起始符（/*、'）和结束符（*/、'）在同一行，如以下注释中，test2 都不与注释的起始符和结束符在同一行。

```
/*test1
   test2
   test3
*/
```

因此不能仅仅依靠上下文判断它们，否则程序将无法有效识别它们。

1.处理多行注释思路如下：

（1）标记开始：当遇到/*时，设置全局变量 flag1 标志为 true，表示进入多行注释状态。

（2）读取注释内容：在多行注释状态中，函数会持续读取字符直到找到注释结束标志*/。读取的字符被存储在 token.comment 数组中。

（3）处理跨行：如果一行结束时仍未找到*/，下一次调用 GetToken 时会继续从下一行的开始处读取字符，直到找到结束标志。

（4）标记结束：当找到*/时，将这两个字符也添加到 token.comment 中，并设置 flag1 为 false，表示多行注释结束。

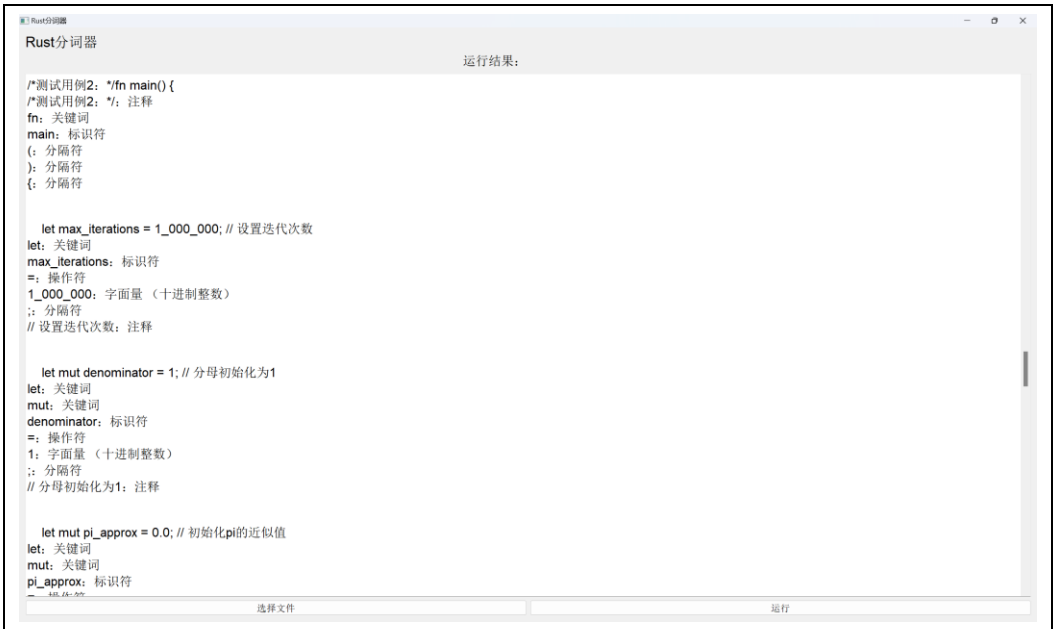
（5）设置词法单元类型：在多行注释状态中，token.ID 被设置为 COMMENT。

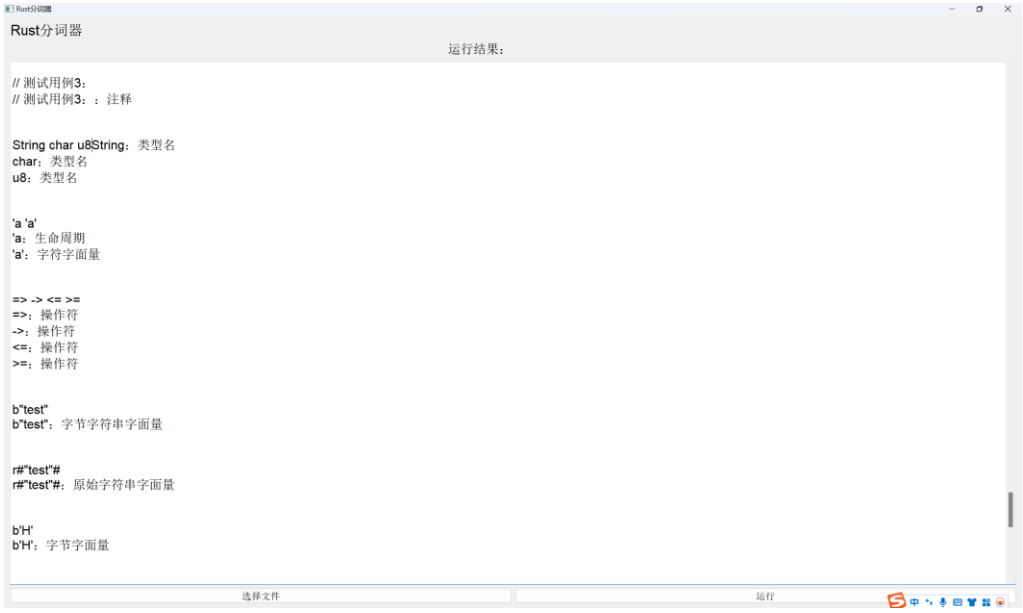
2.处理多行字符串的思路与处理多行注释的思路是一致的，都需要引入标志作为全局变量，来标记多行字符串的范围，此处不再赘述。

（四）测试

（测试用例和结果详见作业文件夹）

测试结果（部分）：





四、实验总结（心得体会）

通过这次实验，我能够将编译原理中的理论知识应用到实际编程中，加深了对词法分析过程的理解。在处理字符串和注释时，我意识到了边界情况的重要性，如跨行的注释和字符串。学会如何处理这些边界情况对于编写健壮的代码至关重要。构建一个能够处理复杂语言结构的分词器，给我带来了成就感。

五、参考文献：

《编译原理及实践》Kenneth C.Louden 机械工业出版社