



**《编译技术》**

**课程实验报告**

姓名：宋泽涛

学院：信息学院

系：软件工程

专业：软件工程

学号：25120222201292

2025年4月11日

1. **实验环境**

PC微机

Windows 操作系统

Turbo C 程序集成环境或 Visual C++ 程序集成环境

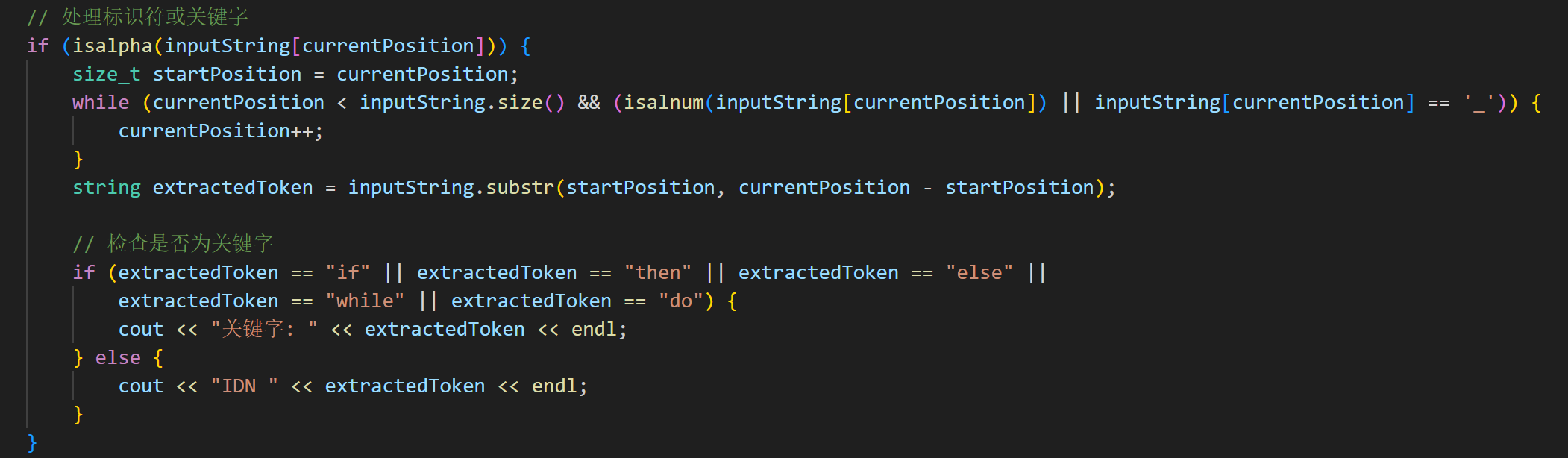
1. **实验目的**

基本掌握计算机语言的词法分析程序的开发方法。

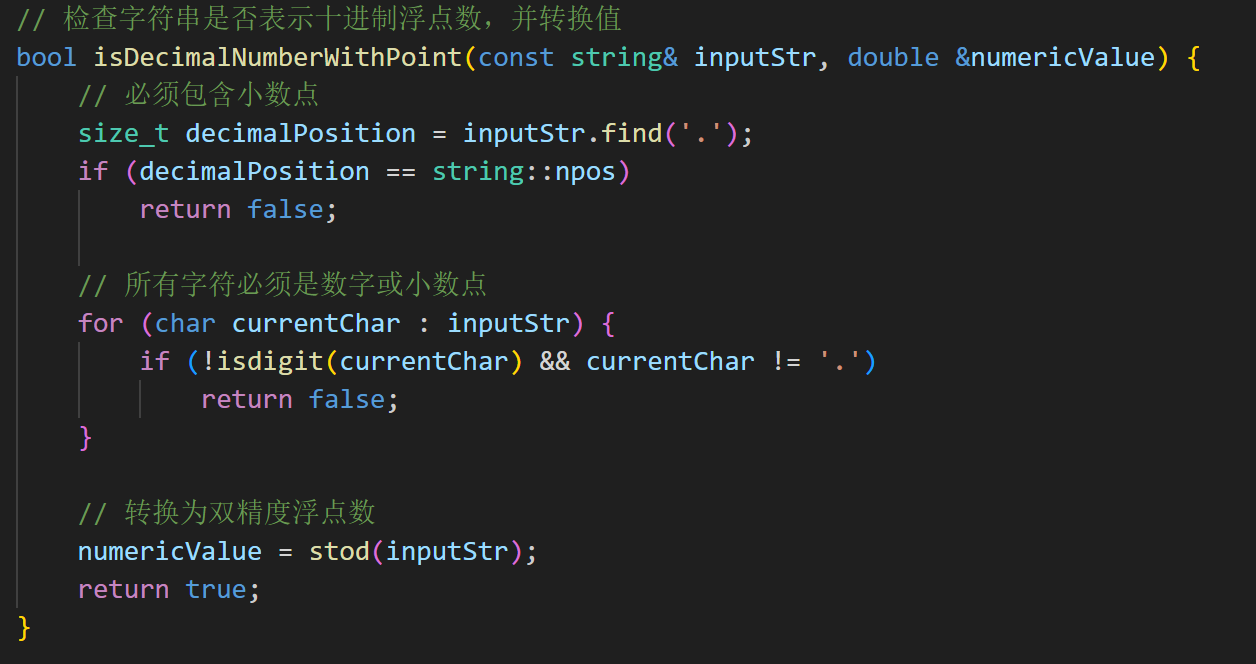
1. **实验内容**

编制一个能够分析三种整数、标识符、主要运算符和主要关键字的词法分析程序。

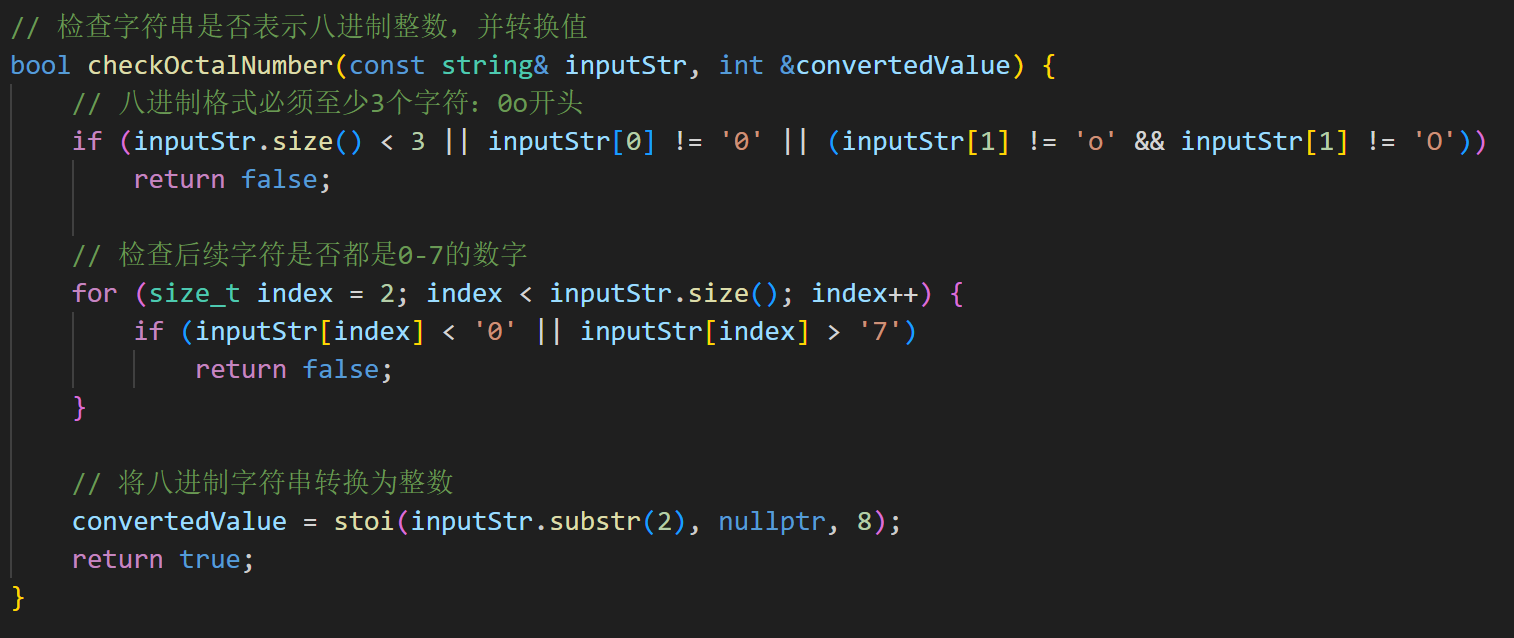
代码部分：



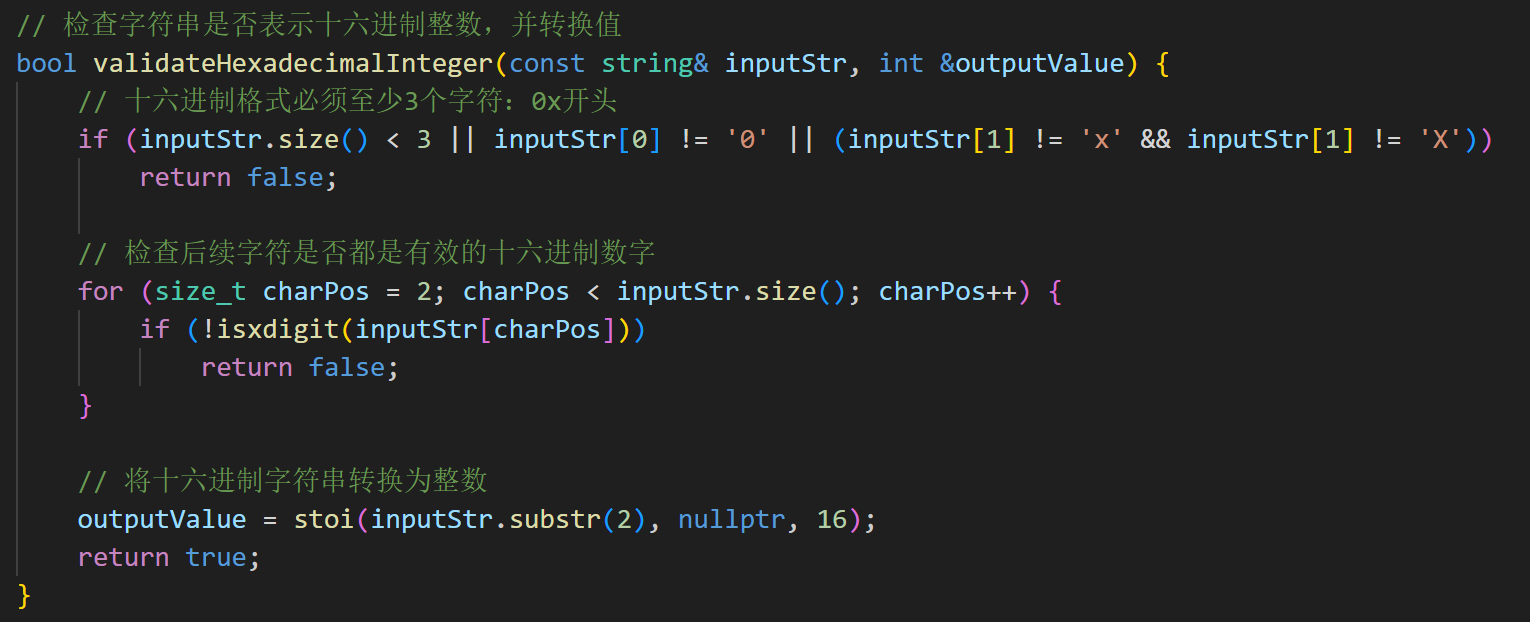
检查字符串是否为标识符或关键字



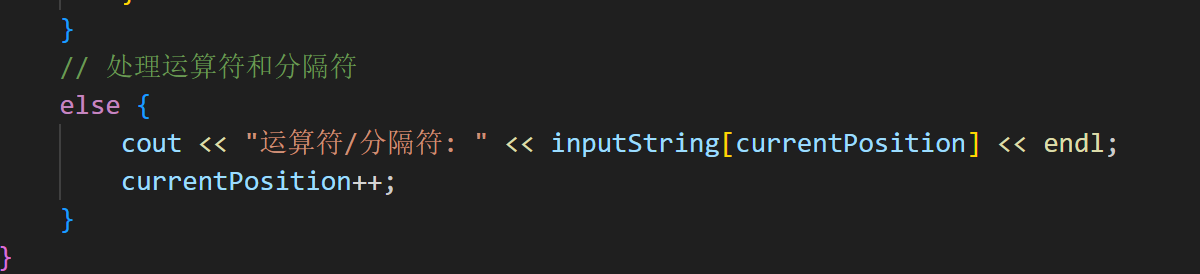
检查字符串是否为十进制浮点数



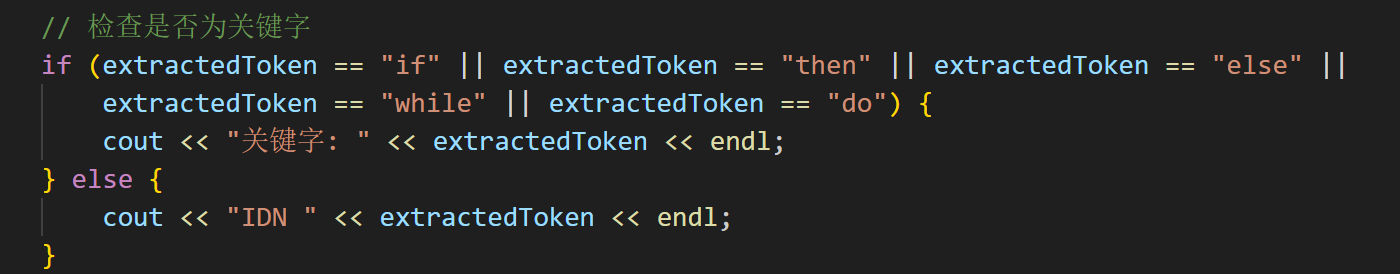
检查字符串是否为八进制整数



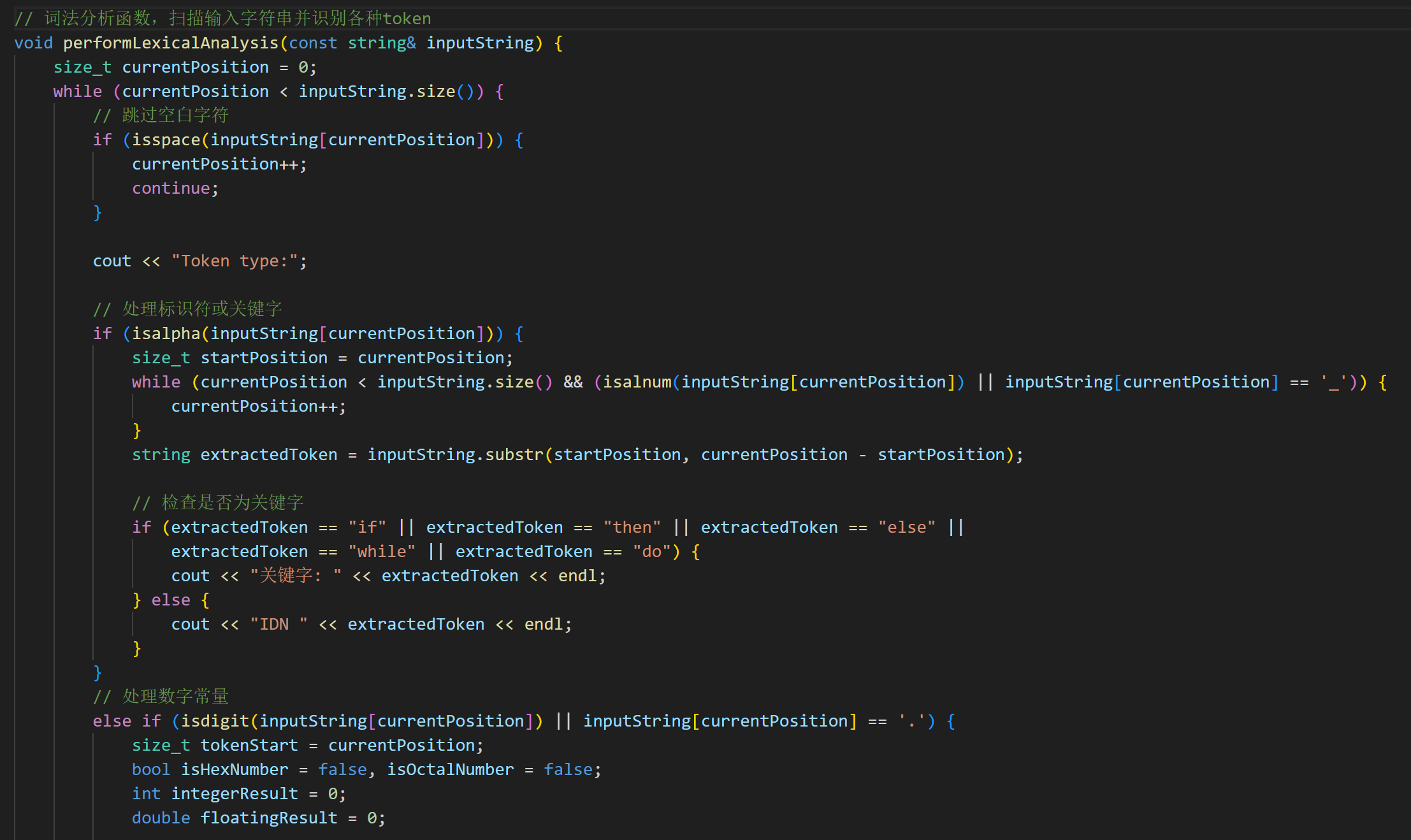
检查字符串是否为十六进制整数

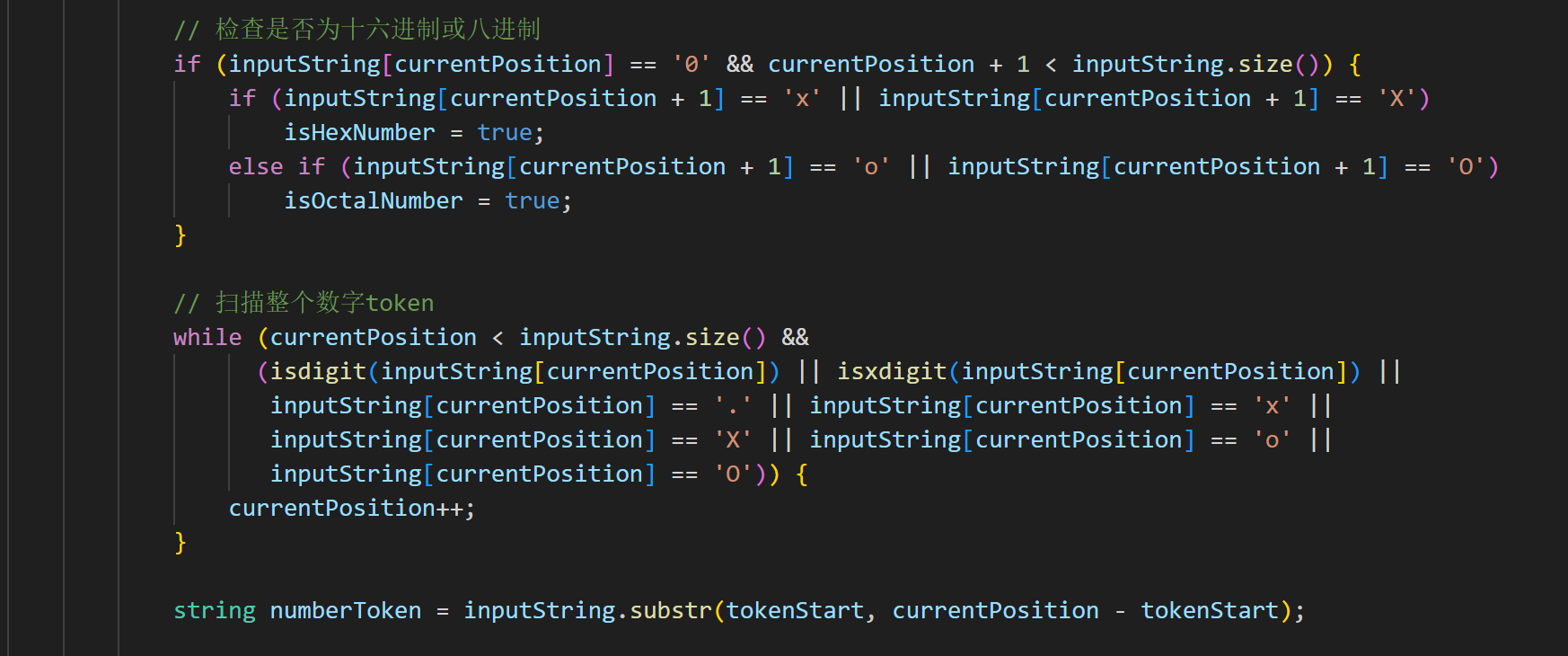


检查字符是否为运算符或分隔符。（当字符既不是字母也不是数字时，将其视为运算符或分隔符并输出）



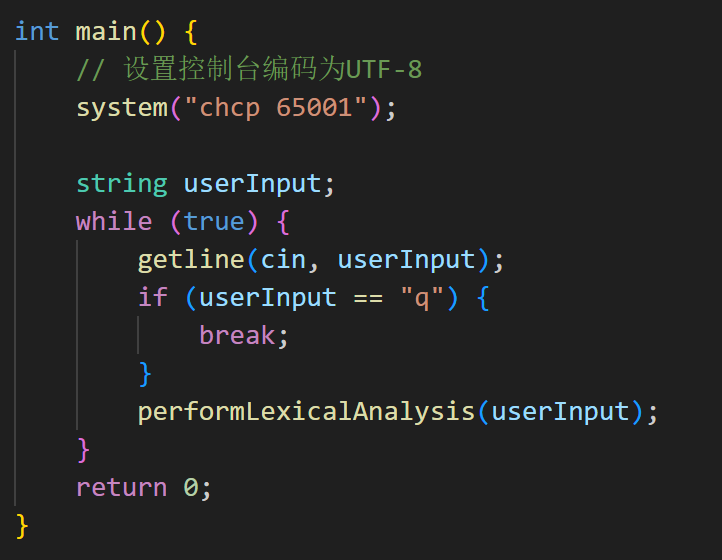
检查字符串是否为关键字。



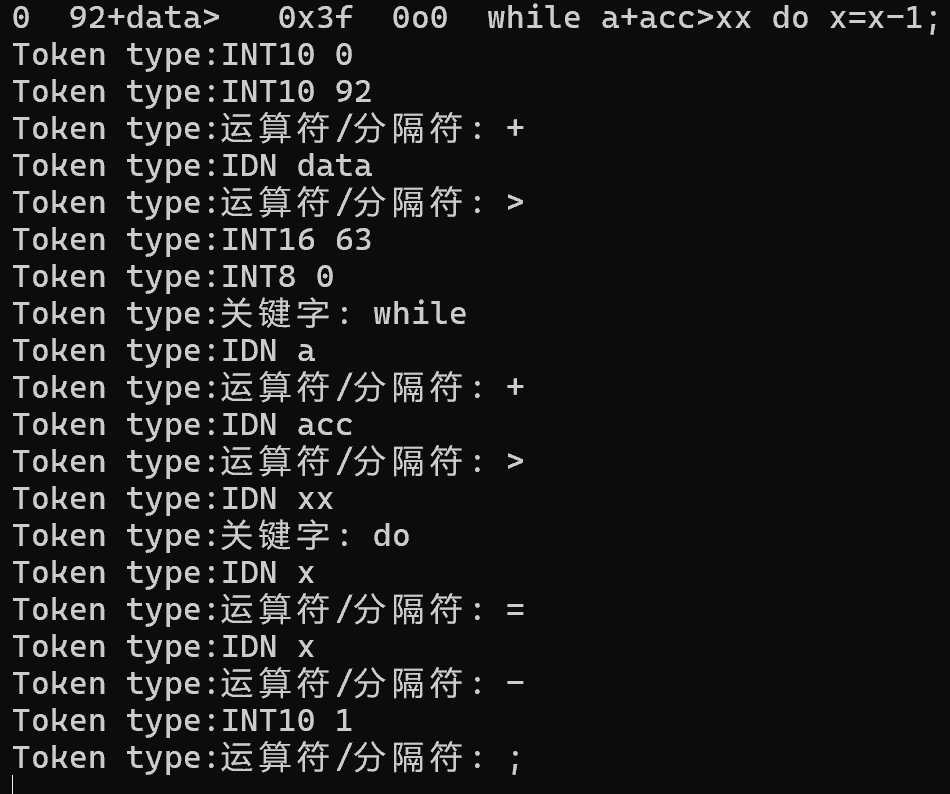


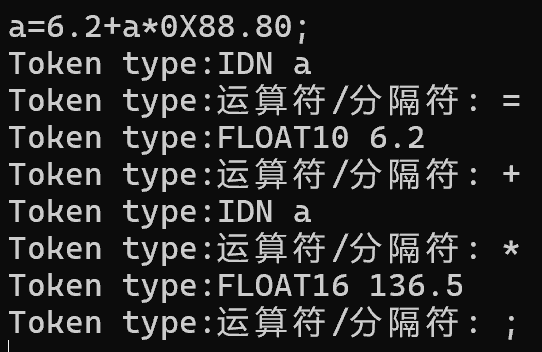


performLexicalAnalysis函数用于对输入的字符串进行扫描和解析。它会遍历字符串中的每个字符，跳过空白字符。根据字符类型，将字符串拆分成不同的标记，并判断这些标记是关键字、标识符、整数、浮点数还是运算符 / 分隔符，最后输出相应的信息



**main 函数**：从用户输入中读取字符串，并调用 scan 函数进行解析。（chcp 65001是用于限定终端提示符为正确的编码）

**测试结果**：  






1. 词法的正规式描述

关键字

关键字集合为 {if, then, else, while, do}，其正规式为：

if|then|else|while|do

标识符（IDN）

标识符以字母开头，后面可以跟字母、数字或下划线，正规式为：

[a-zA-Z][a-zA-Z0-9\_]\*

十进制整数（INT10）

由一个或多个数字组成，正规式为：

[0-9]+

八进制整数（INT8）

以 0o 或 0O 开头，后面跟一个或多个 0 到 7 的数字，正规式为：

0[oO][0-7]+

十六进制整数（INT16）

以 0x 或 0X 开头，后面跟一个或多个十六进制数字（0 - 9，a - f，A - F），正规式为：

0[xX][0-9a-fA-F]+

十进制浮点数（FLOAT10）

包含小数点，小数点前后至少有一个数字，正规式为：

[0-9]\*\.[0-9]+|[0-9]+\.[0-9]\*

八进制浮点数（FLOAT8）

以 0o 或 0O 开头，包含小数点，小数点前后的数字范围是 0 到 7，正规式为：

0[oO][0-7]\*\.[0-7]+|0[oO][0-7]+\.[0-7]\*

十六进制浮点数（FLOAT16）

以 0x 或 0X 开头，包含小数点，小数点前后的数字是十六进制数字，正规式为：

0[xX][0-9a-fA-F]\*\.[0-9a-fA-F]+|0[xX][0-9a-fA-F]+\.[0-9a-fA-F]\*

运算符/分隔符

单个非字母、非数字、非空白字符，假设运算符和分隔符集合为 { +, -, \*, /, (, ), ; }，正规式为：

[\+\-\\*/\(\);]

2. 变换后的正规文法

关键字

设 S为关键字开始符号：

S ->if | then | else | while | do

标识符（IDN）

设 S 为标识符开始符号：

S-> L A

A-> L A | D A | \_ A |ɛ

L-> [a-zA-Z]

D-> [0-9]

十进制整数（INT10）

设 S 为十进制整数开始符号：

S-> D B

B-> D B |ɛ

D-> [0-9]

八进制整数（INT8）

设 S\_{int8} 为八进制整数开始符号：

S-> 0 O C

O-> o | O

C-> [0-7] C |ɛ

十六进制整数（INT16）

设 S 为十六进制整数开始符号：

S-> 0 X H

X-> x | X

H-> [0-9a-fA-F] H |ɛ

十进制浮点数（FLOAT10）

设 S 为十进制浮点数开始符号：

S-> D\* . D+ | D+ . D\*

八进制浮点数（FLOAT8）

设 S 为八进制浮点数开始符号：

S-> 0 O [0-7]\* . [0-7]+ | 0 O [0-7] + . [0-7]\*

O-> o | O

十六进制浮点数（FLOAT16）

设 S 为十六进制浮点数开始符号：

S-> 0 X [0-9a-fA-F]\* . [0-9a-fA-F]+ | 0 X [0-9a-fA-F]+ . [0-9a-fA-F]\*

X-> x | X

运算符/分隔符

设 S为运算符/分隔符开始符号：

S-> [\+\-\\*/\(\);]

3. 状态图

关键字

从初始状态开始，依次匹配关键字的每个字符，如果全部匹配成功则进入接受状态。例如对于 if，初始状态 -> 匹配 i -> 匹配 f -> 接受状态。

标识符

初始状态 -> 匹配字母 -> 匹配字母、数字或下划线（可多次循环） -> 接受状态

十进制整数

初始状态 -> 匹配数字（可多次循环） -> 接受状态

八进制整数

初始状态 -> 匹配 0 -> 匹配 o 或 O -> 匹配 0 - 7 的数字（可多次循环） -> 接受状态

十六进制整数

初始状态 -> 匹配 0 -> 匹配 x 或 X -> 匹配十六进制数字（可多次循环） -> 接受状态

十进制浮点数

初始状态 -> 匹配数字（可选） -> 匹配 . -> 匹配数字（至少一个） -> 接受状态

或者 初始状态 -> 匹配数字（至少一个） -> 匹配 . -> 匹配数字（可选） -> 接受状态

八进制浮点数

初始状态 -> 匹配 0 -> 匹配 o 或 O -> 匹配 0 - 7 的数字（可选） -> 匹配 . -> 匹配 0 - 7 的数字（至少一个） -> 接受状态

或者 初始状态 -> 匹配 0 -> 匹配 o 或 O -> 匹配 0 - 7 的数字（至少一个） -> 匹配 . -> 匹配 0 - 7 的数字（可选） -> 接受状态

十六进制浮点数

初始状态 -> 匹配 0 -> 匹配 x 或 X -> 匹配十六进制数字（可选） -> 匹配 . -> 匹配十六进制数字（至少一个） -> 接受状态

或者 初始状态 -> 匹配 0 -> 匹配 x 或 X -> 匹配十六进制数字（至少一个） -> 匹配 . -> 匹配十六进制数字（可选） -> 接受状态

运算符/分隔符

初始状态 -> 匹配运算符/分隔符 -> 接受状态

4. 词法分析程序的数据结构与算法

数据结构

- 字符数组或字符串：用于存储输入的源代码。

- 状态机：可以用一个二维数组来表示状态转移表，每一行表示一个状态，每一列表示一个输入字符类型（如字母、数字、运算符等），数组元素表示转移到的下一个状态。

- 符号表：可以用哈希表来存储关键字和标识符等信息，方便快速查找。

算法

输入: 源代码字符串 s

输出: 单词序列

初始化:

当前位置 i = 0

状态机初始状态 state = 初始状态

循环 (i < s.length):

读取当前字符 c = s[i]

根据当前状态 state 和字符 c，查找状态转移表得到下一个状态 next\_state

if next\_state 是接受状态:

提取从起始位置到当前位置的子字符串作为一个单词

检查该单词是否为关键字（通过符号表查找）

输出单词类型和单词内容

重置起始位置和状态机状态

else if next\_state 是无效状态:

报错，输入不符合词法规则

else:

更新状态 state = next\_state

i++

思考题

1. 词法分析能否采用空格来区分单词？

词法分析可以部分采用空格来区分单词，但不能完全依赖空格。空格通常可以作为分隔符，用于区分一些明显的单词，如关键字和标识符、整数等。例如，在 if a > 10 中，空格可以将 if、a、> 和 10 分隔开。

然而，有些情况下不能仅靠空格来区分单词：

- 运算符和分隔符：例如 a+b 中的 + 不能用空格分隔。

- 数字和标识符：在某些编程语言中，a1 是一个标识符，不能用空格分隔成 a 和 1。

- 字符串常量：字符串中可能包含空格，如 "hello world"，不能将其拆分成多个单词。

2. 程序设计中哪些环节影响词法分析的效率？如何提高效率？

影响效率的环节

- 状态转移表的查找：如果状态转移表很大，每次查找下一个状态的时间开销会增加。

- 字符读取和处理：频繁的字符读取和比较操作会消耗时间。

- 符号表的查找：如果符号表很大，查找关键字和标识符的时间会变长。

提高效率的方法

- 优化状态转移表：可以采用压缩状态转移表的方法，减少不必要的状态和转移。例如，使用状态合并和最小化算法。

- 字符缓冲：一次读取多个字符到缓冲区，减少磁盘或网络 I/O 操作。

- 符号表优化：使用哈希表等高效的数据结构来存储符号表，提高查找速度。

- 并行处理：对于大规模的源代码，可以采用并行处理的方式，将源代码分成多个部分同时进行词法分析。

**用到的一些cctype函数：**

**1. isalpha**

**功能**：检查字符是否为字母（大写或小写）。

**原型**：int isalpha(int c);

**返回值**：如果 c 是字母（A-Z 或 a-z），返回非零值（真）；否则返回零（假）。

**2. isalnum**

**功能**：检查字符是否为字母或数字。

**原型**：int isalnum(int c);

**返回值**：如果 c 是字母（A-Z 或 a-z）或数字（0-9），返回非零值（真）；否则返回零（假）。

**3. isdigit**

**功能**：检查字符是否为数字。

**原型**：int isdigit(int c);

**返回值**：如果 c 是数字（0-9），返回非零值（真）；否则返回零（假）。

**4. isxdigit**

**功能**：检查字符是否为十六进制数字（0-9, A-F, a-f）。

**原型**：int isxdigit(int c);

**返回值**：如果 c 是十六进制数字（0-9, A-F, a-f），返回非零值（真）；否则返回零（假）。

**5. isspace**

**功能**：检查字符是否为空白字符（空格、制表符、换行符等）。

**原型**：int isspace(int c);

**返回值**：如果 c 是空白字符，返回非零值（真）；否则返回零（假）。

**6. islower 和 isupper**

**islower**：

**功能**：检查字符是否为小写字母。

**原型**：int islower(int c);

**返回值**：如果 c 是小写字母（a-z），返回非零值（真）；否则返回零（假）。

**isupper**：

**功能**：检查字符是否为大写字母。

**原型**：int isupper(int c);

**返回值**：如果 c 是大写字母（A-Z），返回非零值（真）；否则返回零（假）。