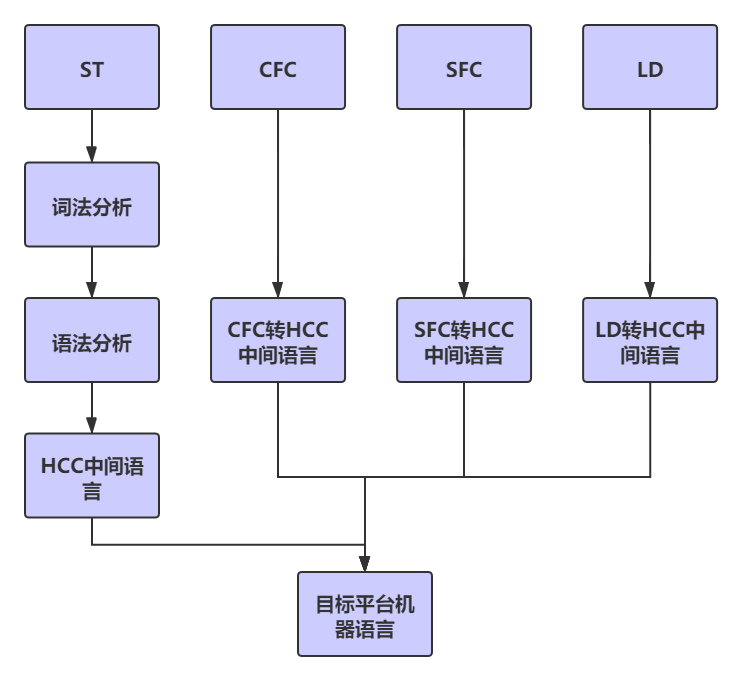
**ST语言转C语言实现方案调研报告**

# 调研背景

M6在实现时，直接将ST语言翻译成目标平台的二进制代码；M7在设计时，为了更方便的支持更多的库，计划将ST语言翻译成C语言，然后通过LLVM/clang的二次开发翻译成目标平台的二进制代码。因此，我们需要基于ST语言的一些特点，调研ST语言转C语言的实现方案。

# M6编译简介



# 调研内容

调研/研究比较成熟的语言转换的实现方案，结合项目现状和特点，提出方案。

# 验证标准

1. **转换方案比较成熟，实现风险小，可以付诸实施**
2. **适合M7项目的特点，能够很好的支持M6实现的ST语言的特性**
3. **在验证过程中，需要采用最小原型的方法进行验证**
4. **工作量尽可能小**
5. **可维护性比较好**

# 验证过程

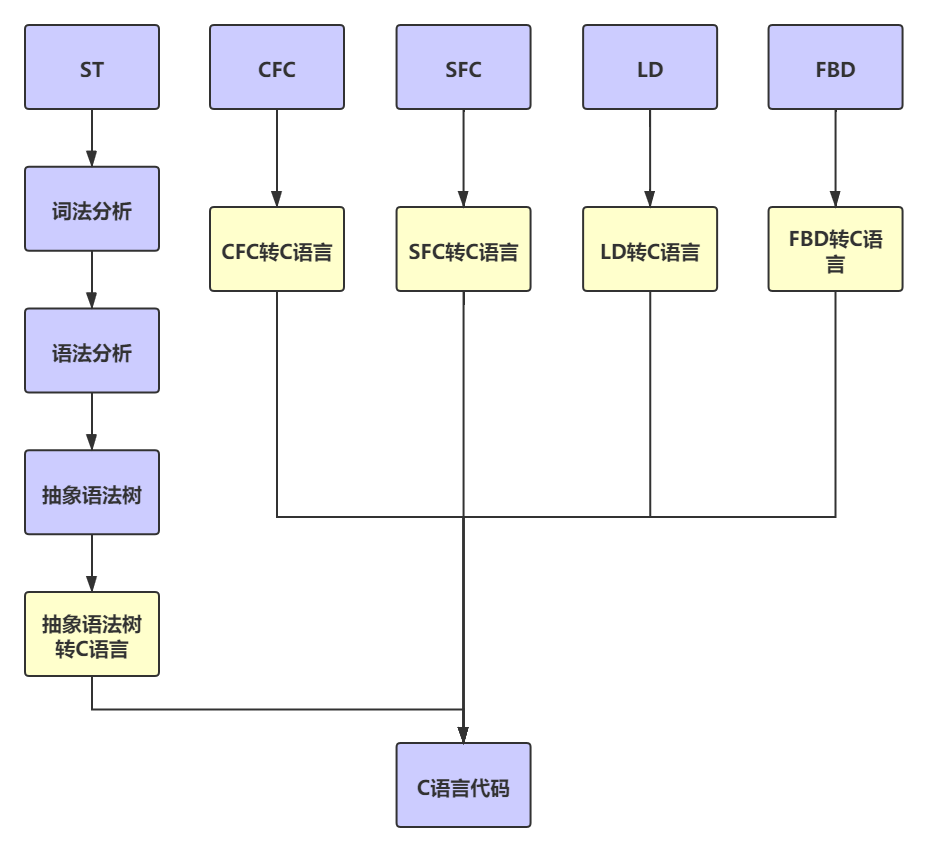
在调研过程中，主要考虑了以下两种方案：

1. **基于正则表达式匹配的方式进行ST语言的转换**
2. **基于抽象语法树(AST)的方式进行转换**

第一种方案，通过正则表达式匹配ST语言的语法元素，通过替换的方式转换成C语言，考虑到编写代码的**语法错误**，**灵活性**，**支持的范围**（只能支持形式化语言，图形语言需要另外处理），以及**复杂语法**如指针和数组的可支持性，感觉可行性比较差，直接废弃，没有进行进一步的尝试。（目前一些代码编辑器如notepad++，Visual Studio Code在代码高亮显示时，采用类似正则表达式匹配的方案）

第二种方案，通过词法分析和语法分析，生成抽象语法树，通过**遍历抽象语法树，对相应的语法元素进行处理，将抽象语法树翻译成C语言**。这种方案，**ST语言可以很好的支持。**

虽然各种图形化语言存在差异，都是都可以分解为基本的程序结构（顺序，选择，循环）和语句。



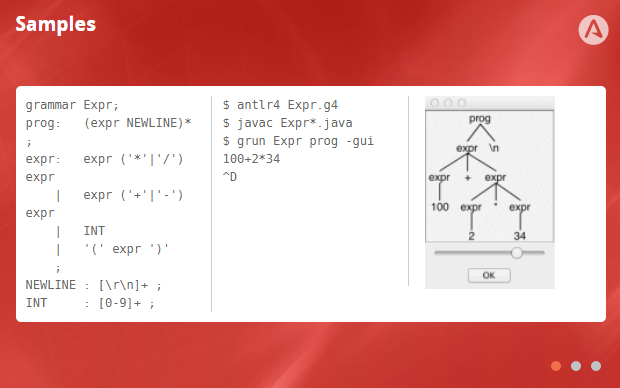
# ST语言调研方案

## 使用ANTLR进行词法和语法分析

### 简介

ANTLR(ANother Tool for Language Recognition)是一款开源的编译器框架和工具集，通过输入词法和文法文件（合称**语法文件**），生成词法分析和语法分析的代码等，并且可以对语法树进行可视化显示。

Facebook开源的大数据sql查询引擎presto就是用的ANTLR做底层SQL语法解析；开源数仓工具hive也使用ANTLR进行SQL的解析。



（整数的四则混合运算，可以使用括号）

### 词法分析/语法分析/抽象语法树生成

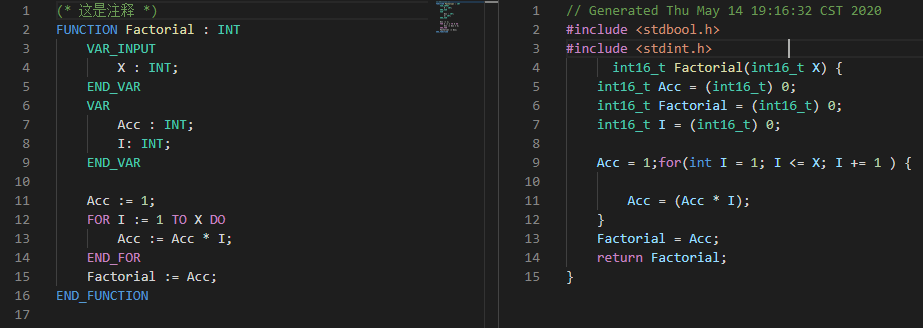
使用开源的IEC词法和文法文件，通过下面命令行生成C++代码：

*java -jar antlr-4.8-complete.jar -Dlanguage=Cpp -listener -visitor -o generated/ -package ieclang IECLexer.g4 IECParser.g4*

### 代码转换

开源项目verifaps-lib中包含一个从ST语言到C语言转换的小工具。

### 演示



## 基于M6的词法和语法分析

### Lex简介

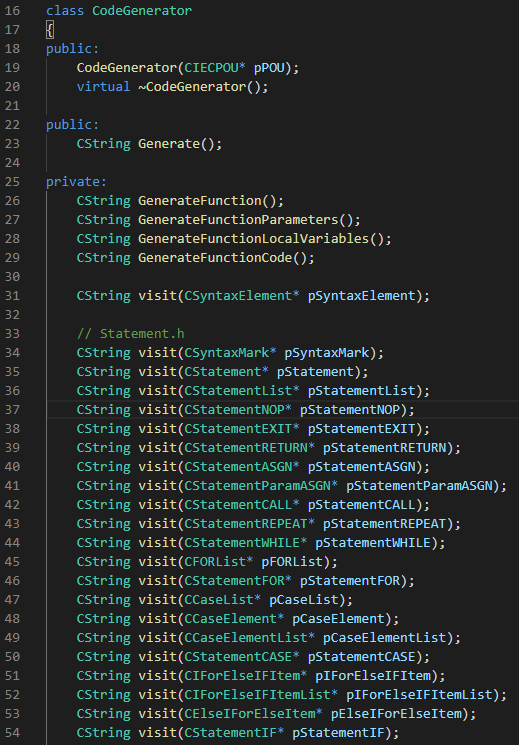
Lex是一个生成词法分析器的工具，词法文件是由一系列的规则组成，每个规则包括相应的动作。Lex接收到文件或文本形式的输入时，它试图将文本与规则进行匹配。它一次读入一个输入字符，直到找到一个匹配的模式。如果能够找到一个匹配的模式，Lex 就执行相关的动作（可能包括返回一个标记）。 另一方面，如果没有可以匹配的规则，将会停止进一步的处理，Lex 将显示一个错误消息。

### 词法分析/语法分析/抽象语法树生成

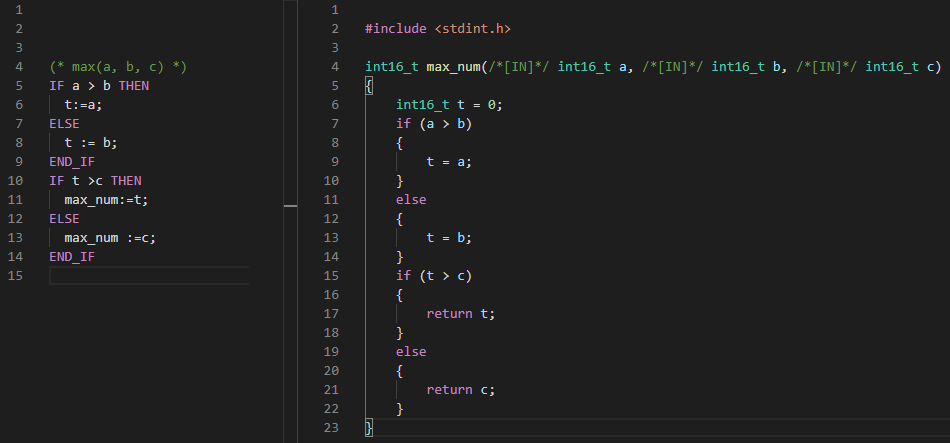
1. M6 ST语言的词法分析通过编写Lex词法文件，通过Lex工具自动生成词法分析代码（详见LexForIEC.l和LexForIEC.c）
2. M6 ST语言的语法分析从词法分析中获取Token，手动编写代码进行语法分析（详见类CIECParse），经过语法分析，生成抽象语法树

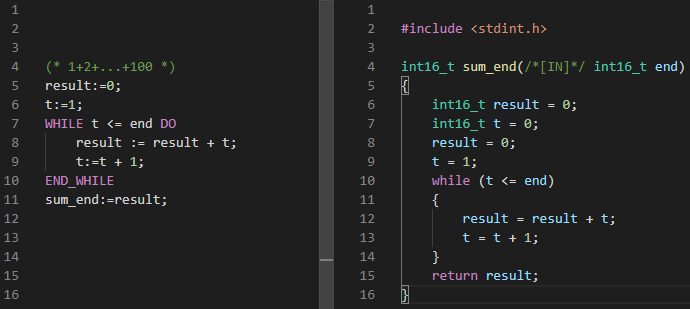
### 代码转换

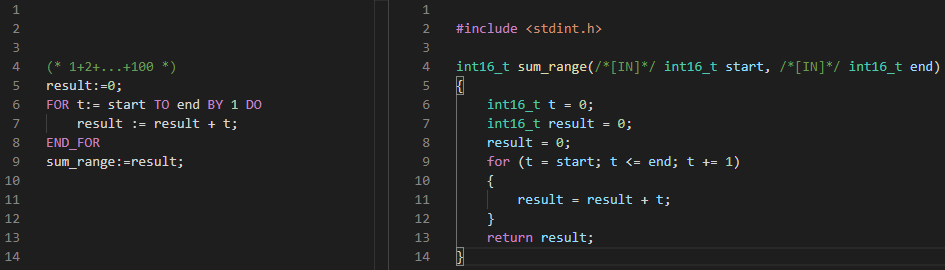
编写代码遍历语法分析生成的抽象语法树，针对不同的语法结构生成相应的C语言代码。部分代码如下：

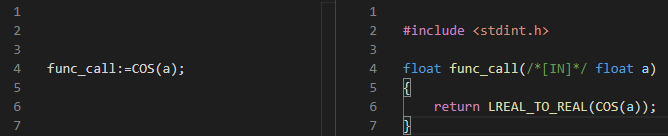


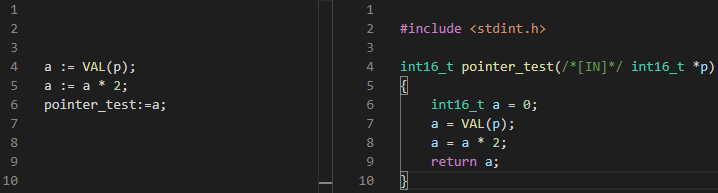
### 演示

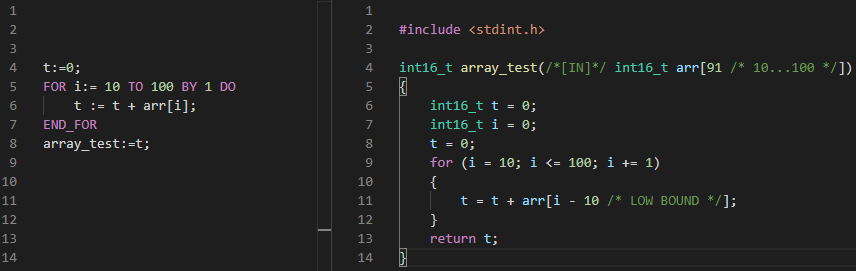












## 总结

|  |  |  |
| --- | --- | --- |
|  | **基于ANTLR** | **基于M6** |
| **技术风险** | 风险大，毕竟之前没有使用过ANTLR，从技术上讲有一定的不确定性 | M6的词法分析和语法分析经过实践检验，比较成熟，通过将编译器后端修改为代码转换相对风险比较小  √ |
| **工作量** | 1. 使用现成的词法和语法文件进行修改，与M6 ST语言的实现保持一致 2. 遍历ANTLR生成的语法树，针对语法元素生成代码   √ | 1. 将M6的词法分析和语法分析代码中使用到MFC的使用STL进行替换，并且保持代码逻辑一致 2. 遍历ANTLR生成的语法树，针对语法元素生成代码   √ |
| **可维护性** | 词法分析和语法分析的代码通过ANTLR自动生成，只需要维护词法文件和语法文件  √ | 需要维护M6 ST语言词法分析和语法分析相关的代码，M6 ST语言的词法分析使用lex生成，可维护性还好；语法分析使用手动编写代码的方式实现，维护工作量较大；后续加入新的语法比使用ANTLR复杂 |
| **IEC标准支持程度** | 找到的开源的ANTLR词法和语法文件对IEC标准支持比较全  √ | 目前IEC标准支持存在欠缺，如果要完全支持标准，需要手动编写代码补充语法分析 |

# 调研结论

**基于抽象语法树进行ST语言到C语言的转换，复用M6 ST语言的词法分析/语法分析/抽象语言树生成相关的代码。**

# 讨论

## HCC中间代码转C语言

|  |  |  |
| --- | --- | --- |
|  | **基于M6词法和语法分析转C语言** | **基于HCC中间代码转C语言** |
| **可行性** | 可行  √ | 可行，两者都是语法树，虽然不在一层  √ |
| **难度** | 适中，主要复杂性在于细节比较多，需要熟悉IEC语言和C语言  √ | 稍大，HCC中间代码细节性内容稍多 |
| **工作量** | 1. 编码实现ST语言语法树到C语言的转换 2. 其它图形语言需要转换为ST语言语法树 | 1. HCC语法树直接转C语言，其它图形语言已经转为HCC中间代码   √ |
| **性能** | 比后者少一个生成HCC中间代码的过程  √ |  |
| **可维护性** | √ | HCC编译器已经不再使用，是否需要保留HCC中间代码。（整体移除，还是部分移除） |