Compiler

2015/12/23

## 概述

|  |  |
| --- | --- |
|  | 这个编译器主要实现以下功能：   1. 从文件中读取程序 2. 对程序进行词法分析，不符合词法规则的将会报错 3. 从词法分析输出得到的结果进行语法分析，构建语法树 4. 扫描语法树，建立符号表 5. 扫描语法树，生成llvm-IR 6. 语义错误检测在符号变建立和IR生成的过程中同步完成 7. 执行生成的llvm-IR获得结果 |

## 词法分析

|  |  |
| --- | --- |
|  | 词法分析主要使用flex，使用regular expression来表示合法的字和符号，对于不合法的，报错。关于词法的定义见grammar.pdf。 |

## 语法分析

|  |  |
| --- | --- |
|  | 语法分析主要使用bison，它读入词法分析的的子和符号，采用CFG语法，分析器用的是LALR（1），对于应该reduce的操作，new一个树的节点表示这个reduce，并在节点中记录节点的信息（包括行数，节点类型等等）。  如果程序成功accept，说明程序语法是正常的，语法树就成功建立了，建立好语法树就可以进入下一步工作了。关于语法的定义见grammar.pdf。 |

## 符号表的建立

|  |  |
| --- | --- |
|  | 编译器的后端首先根据ast树信息建立符号表。对于符号表的设计主要包含全局变量表，变量表，类型表(变量表和类型表的区别是变量表为基本类型的变量，而类型表为定义数据结构（数组、类）的变量)，函数表（包含函数参数、变量、类型信息）。 |

## 语义检查

|  |  |
| --- | --- |
|  | 语义错误的检查在建立符号表和IR生成的过程中同步完成。  主要处理的错误有类型错误，参数不匹配，变量重定义等。 |

## LLVM-IR生成

|  |  |
| --- | --- |
|  | 根据节点类型进行代码分析，并对应LLVM-IR的具体语法将ast树信息转化为IR代码。  若无语义错误，此处输出的IR代码即为LLVM可执行代码，否则是语义错误信息。 |

## 编译器用法

|  |  |
| --- | --- |
|  | 运行程序，打开文件  运行完成之后会生成llvm文件，比如输入：  qsort.mylang会生成qsort.ll文件  运行llc qsort.ll  clang qsort.s -o qsort  ./qsort  就可以看到运行结果了。 |

## 测试用例

|  |  |
| --- | --- |
|  | 测试用例主要为以下文件：  helloworld.mylang hello world程序  qsort.mylang 快排  queen.mylang 8皇后问题 |
|  |  |