

# 编译原理大作业

---

## 实验目标

我们要实现一个c语言的编译器。用Lex开发该语言的词法分析器，用YACC开发该语言的语法分析，用Illum生成中间代码和目标代码，最终生成目标代码（调用Illum的api可以把Illum中间代码编译成目标机器的汇编代码，我们先不决定汇编成那种语言）

## 实验要求

实验要求：

### 1、提交实验报告

编译器包含词法分析、语法分析、语义分析、代码生成、代码优化、运行环境等阶段和环节。报告中要**包括词法分析部分的正规表达式描述、数据结构、原理以及实现，语法分析部分的文法描述、数据结构、原理以及实现，语义分析的方法描述，中间代码的格式、数据结构描述以及中间代码生成的实现，目标代码的生成。**

实验报告中要明确组员分工情况。

### 2、提交原代码

## 实验环境

- (1) Linux 2.6以上版本
- (2) GCC3.4以上版本
- (3) Bison 2.2以上版本
- (4) Flex 2.5.33以上版本
- (5) Illum 14左右（因为不同版本的Illum的api不一致）

### 2. Ubuntu

#### ◦ 安装Flex & Bison

```
sudo apt-get update && sudo apt-get upgrade && sudo apt-get install flex  
bison
```

#### ◦ 安装LLVM C++ API

```
sudo apt-get install llvm-14
```

# 实验任务

我们要实现的c语言文法定义：

- 所有C语言基本语句。包括 `if`, `for`, `while`, `do`, `switch`, `case`, `break`, `continue`, `return`。
- 所有C语言表达式。包括括号 `()`, 数组下标 `[]`, `sizeof`, 函数调用, 结构体 `->`, `.`, 一元 `+`, 一元 `-`, 强制类型转换, 前缀 `++`, 前缀 `--`, 后缀 `++`, 后缀 `--`, 取地址 `&`, 取内存 `*`, 位运算 `&`, `|`, `~`, `^`, 逻辑运算 `||`, `&&`, `!`, 比较运算 `>`, `>=`, `<`, `<=`, `==`, `!=`, 算术运算 `+`, `-`, `*`, `/`, `%`, 移位运算 `<<`, `>>`, 赋值语句 `=`, `+=`, `-=`, `*=`, `/=`, `%=`, `<<=`, `>>=`, `|=`, `&=`, `^=`, 逗号表达式, 三元运算符 `?:`。
- 类型系统。基本数据类型包括 `bool`, `char`, `short`, `int`, `long`, `float`, `double`, `void`。复杂数据类型 `array`, `struct`, `enum`, `union`。支持 `typedef`。
- 递归式结构体/共用体。结构体内可以定义指向自己类型的指针, 从而实现链表。
- 指针类型。支持任意类型的指针类型, 例如 `int ptr`, `struct{int x, y;} ptr`。并且支持 `&`, `*`, `->` 等指针运算。
- 类型转换。我们的编译器严格按照C语言的类型转换机制设计。包括隐式类型转换, 例如 `int+float->float`, `pointer+int->pointer`; 强制类型转换, 例如 `(int)1.0`, `(double ptr)malloc(8)`。
- 左值和右值。我们的编译器可以编译C语言支持的任意表达式, 例如 `*p<<=(c?a:b)[3]=st->x+sizeof(double array(5))`;。运算符优先级参考 [C Operator Precedence](#)。
- 可变长参数列表。例如 `void sum(int n, ...)`;
- 函数先声明后定义。编译器会检查声明和定义的类型是否一致。若函数只有声明, 则由链接器负责链接外部函数。
- 调用C标准库的函数。只要提前声明即可。例如 `void ptr malloc(long)`, `int printf(char ptr, ...)`。
- 符号表作用域。我们的编译器允许在 `if`, `for`, `while`, `do`, `switch` 以及语句块 `{}` 内定义变量, 并且可以覆盖外层作用域的重名变量。变量的作用域只在语句块内。
- 编译优化。支持 `-O0`, `-O1`, `-O2`, `-O3`, `-Os`, `-Oz` 优化选项。

南大编译原理实验语法或者参考资料1

0.找zju前几届大作业实现, 运行一下, 看一下他们怎么实现

1.词法分析和语法分析

lex parser: 郑祎豪、张涵智

2.语义分析

潘宇轩、郑祎豪

3.看llvm文档

ddl: 5.8

## 参考资料

ps:大家可以参考实验目标、测试样例，不要直接参考代码

1.[https://github.com/yihzheng258/2023\\_foc\\_complier](https://github.com/yihzheng258/2023_foc_complier) (查重警告，慎重参考)

2.南大编译原理实验文档 (非常详细)

[编译原理 \(2023年春季\) \(nju.edu.cn\)](http://nju.edu.cn)

3.llvm学习资料

[LLVM入门笔记（一）：环境搭建和一些基本概念 - 知乎 \(zhihu.com\)](https://zhuanlan.zhihu.com/p/111111111)

llvm文档非常复杂，建议早点看，最后中间代码、优化、最终代码生成的任务也是三个人一起的（大概）