Lab03-IR Generation

> 实验名称：NJU-编译原理(2022秋)-Lab03-中间代码生成(Intermediate Representation Generation)
> 实验人员：201220096 钟亚晨
> 完成日期：2022/11/18
> 更多实现细节可见个人博客文章：(pawx2's Blog)【密码：`chcp65001`】

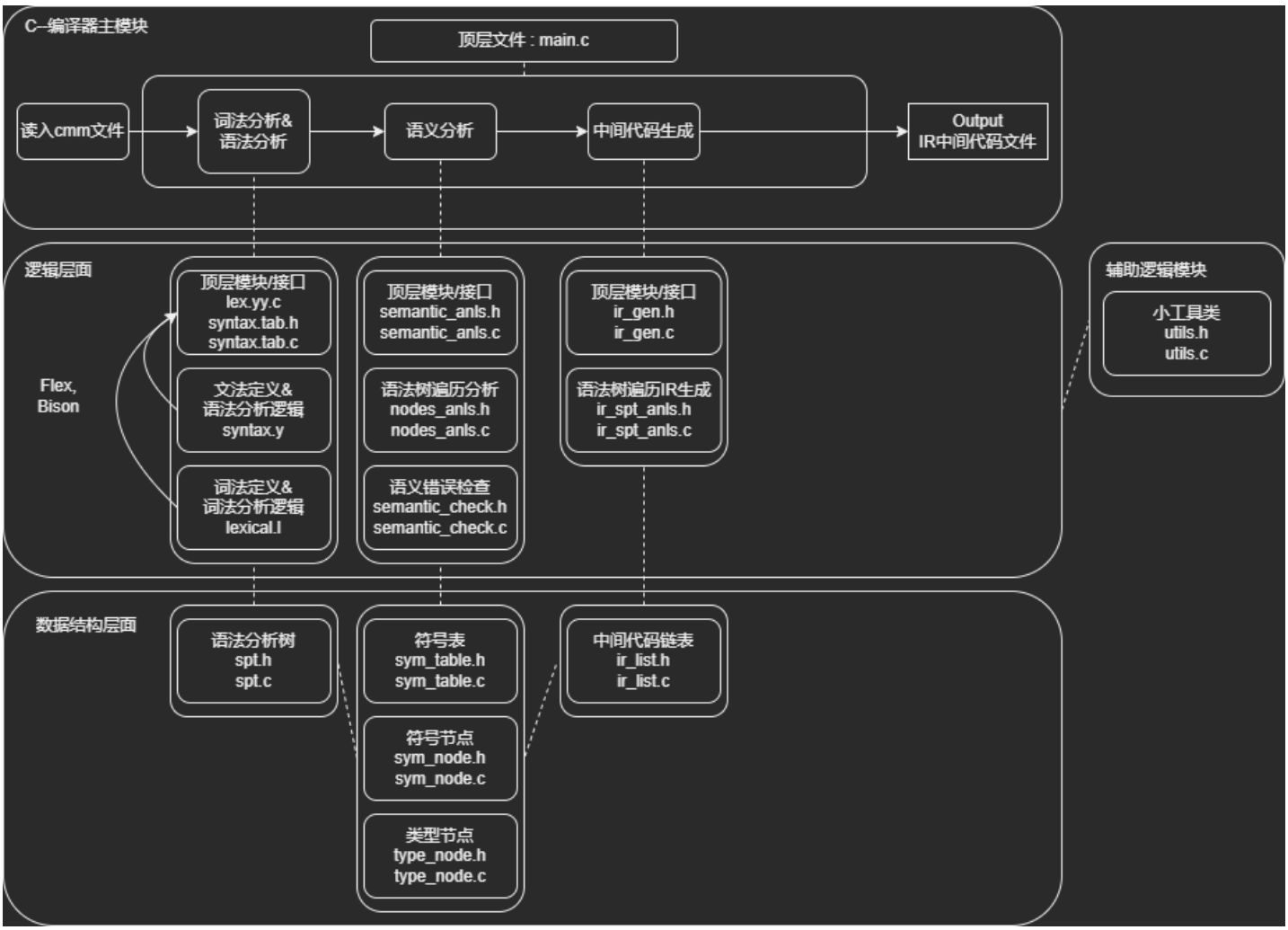
功能实现

- 实现了`实验三--中间代码生成.pdf`中要求的所有**基本内容**、**拓展内容**；
- **通过了OJ平台上的所有测试用例**；

实现思路

- 由于实验说明及指导中给出了大部分明确的语义动作，并且**Label主要作为继承属性**，从而避免回填。因此通过**函数递归、相互调用并组织好顺序结构的代码**即可完成中间代码的生成。

程序架构图



实现亮点

数据结构方面:

中间代码-变量：每个中间变量均为**一个节点**，其定义如下，各个属性值说明以注释形式给出

```
// ./ir_gen/ir_spt_anls.h
typedef struct IrNode{
    char* name; // 中间变量的名字 : temp<序号>
    int if_addr;// 1-中间变量为一个地址; 0-中间变量为一个值
    enum TypeType ntype;// 同类型节点，分为STRUCT_T、BASIC_T、ARRAY_T，用于指示成员u的使用
    union{
        SymNode* struct_def;// 当ntype为STRUCT_T时有效，指向结构体定义
        TypeNode* array_def;// 当ntype为ARRAY_T时有效，指向TypeNode类型节点(数组类型定义)
    }u;
}IrNode;
```

- `name`：中间变量名字，主要用于**作为参数构建中间代码语句**；
- `if_addr`：指示中间变量存储的是地址还是值，主要用于**指示构建中间代码语句时哪些需要进行指针解引用**；
- `union u`：为存储结构体/数组地址的中间变量提供定义信息，**在获取对应成员时可通过类型定义方便地计算偏移量**；

中间代码-语句：每条中间代码语句为**一个节点**，通过**双向链表**进行组织

```
// ./data_structure/ir_list.h
typedef struct IrStmt{
    enum IrType ir_type;// 指定语句类型，依据该类型用指定格式进行打印
    int param_num;      // 参数数目
    char** params;      // 所有参数
    struct IrStmt* next;
    struct IrStmt* prev;
}IrStmt;
```

逻辑实现方面

整体实现逻辑基本与实验指导相同，但**主要依靠代码顺序结构与递归**，基本绝大多数的中间代码生成逻辑为：顺序调用子部分，子部分生成一个中间量存储结果供调用者使用。

以中间变量节点作为连接各个部分的桥梁（缺点：实现复杂度大幅降低，但产生较多冗余变量，而该部分呈现极强的局部性，在后续代码优化中易于处理）：

Trick01：使用**带参数的宏、C语言-不定参函数**
极大地**简化了代码实现、对通用变量标识进行了统一**：

- `./data_structure/ir_list.c, ir_list.h` 中的 `IrStmt* new_ir_stmt(int num, ...);` 函数；
  - 相比于为每一种语句编写一个构造函数，工程量大幅降低
- `./utils/utils.h` 及散步在其它文件中的宏定义；

Trick02：选用了"**语义分析与中间代码生成相分离的**"设计方向，模块化强：

- `./node_anls/` 中存放了**语义分析模块**的所有代码；
- `./ir_gen/` 中存放了**中间代码生成模块**的所有代码；

实验感想

- 本次实验执行了上次实验的启发与复盘结论：
  1. 在开始实验前进行了适当的"思想实验"，通过在纸上**自顶向下**地对项目进行**由粗到细**的分析与调整，未出现像先前一样需要整体调整框架的大工作量的错误，**设计的数据结构与程序框架总体是从头用到尾**且**各个模块划分更为紧凑。**
  2. 对gdb等相关调试工具进行了学习了解，面对"段错误"**得心应手，游刃有余。**
  - 

- 整个课程的实验设计思路目前似乎是：从Lab01->Lab03，**实际能够表示的语言范围在逐步缩小，后续为前序的子集。**因而回过头来思考，**自己前序实验更多地应该采取"渐进式开发"**，使用条件宏控制相关功能的开启与否，而非一步到位高度集成基本功能和拓展功能。（即：**基础功能为主线任务，拓展功能为支线任务**，且渐进开发对于流程的把握应当更加有效）
- 在程序调试方面仍需继续学习，通过对gdb初步使用，一方面大幅提高了debug效率，另一方面，打消了已久的"面对bug"心里没底，不知从何开始的难受状态😫。**会调试程序与会写程序同等重要**，继续加强对gdb、vscode调试工具的学习💪。
- 理论知识&程序设计方面：
  1. 通过实验逐步完成C--编译器的各个模块，还加深了自身对于**编译原理理论知识**的理解，特别是语法分析树，在截至目前，基本是贯穿了三个实验阶段，并为语义分析、中间代码生成提供了完备的信息，相比于理论学习中对其无感，通过实践反而感慨其重要性以及**词法、文法、中间代码规范设计的巧妙。**
  2. 截至目前，编译原理课程的项目量已经超过了以往其它课程的大项目代码量。而由于实验材料充足，以及设计较为完备，整体而言体验是很好的。通过阶段报告复盘、代码重构优化，自身**对于项目规模的把控能力应该相较以前有了一定的提升**，特别是**模块划分、接口设计以及大规模下代码规范。**