编译原理实验二报告

胡俊豪 181240020@smail.nju.edu.cn

1 功能

本实验完成了实验手册中所描述的所有功能，并通过了所有测试样例。

1.1 测试说明

该程序尽量只报告最本质，最底层的错误。最底层指的是，在搜索语法树的过程中，在树的底层发现的错误。这个错误层层上报，会抑制上层的错误（也可以选择汇报上层错误，此时只需要在 common.h 中将 REPORT_ALL_ERROR 从 2 改为 0）。当 REPORT_ALL_ERROR 为 0 时，我们会汇报所有错误，各种各样的嵌套错误等等。

为了完成实验额外的要求（2.1，2.2，2.3），我们舍弃了部分规约（假设）。但需要注意的是，尽管我们舍弃了这些规约，这并不影响我们完成实验所要求的十九个错误类型检测。

第一个舍弃的假设是：匿名结构体都有一个隐藏的名字。因为我们只考虑结构等价，不实现名等价，在别的地方也用不到匿名结构体的隐藏名字，所以我们没有为每一个匿名结构体分配隐藏名。第二个舍弃的假设是，不同结构体中的域必须互不重名。这显然是为不需要完成额外要求的同学设计的假设，但在额外要求里并没有把这个规约去掉，显得很不合理。变量拥有作用域的时候，当然是可以允许不同结构体中，存在相同名字的变量的，故我们舍弃这个规约。

2 如何编译

本实验的编写测试环境，文件夹结构，Makefile 内容均与手册上一致，只需要在 Code 文件夹下使用命令 make，就可以得到一个名为 parser 的可执行文件。输入"./parser testfile.cmm" 即可完成对文件的解析。

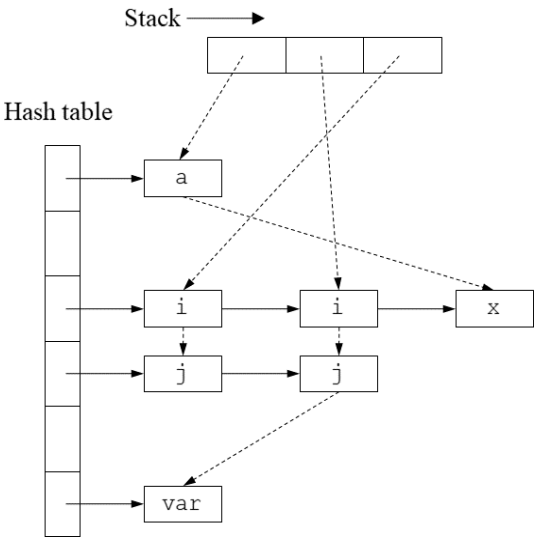

图 1: 符号表架构

3 设计特色

3.1 符号表架构

符号表整体架构如图 1，与实验手册中描述的，能够支持多层作用域的符号表架构一致，采用 imperative style。在符号表的每一个 entry（即图中的每一个 block）中，我们同时记录了 hash_pos 和 stack_pos。hash_pos 能帮助我们定位到这个 entry 的 name 通过 hash 映射后，会达到 hash 数组的哪一个格子；stack_pos 告诉我们，这个 entry 是属于分层作用域中的哪一层。通过比较 stack_pos 就可以比较两个变量所属作用域的的内外层关系。

3.2 类型数据结构

类型数据结构架构图如图 2，基本思路和实验讲义中一致。我们对这个数据结构做了一部分额外定义与修改。其中当 kind 等于 BASIC 的时候，我们定义 basic = 1 时，表示的时 int 类型；basic = 2 时，表示的是 float 类型。最后，为节省空间，我们把 array 和 structure 均改为了指针类型。

3.3 结构体

对于每一个 entry，基本存储的内容有，name，type。在符号表中，我们不仅存储所有定义的变量，我们还存储所有定义过，并且有名字的结构体。name 同样存储这个结构体的名字，type 存储这个结构体定义的 type。在表中，对于变量定义和结构体定义我们一般不

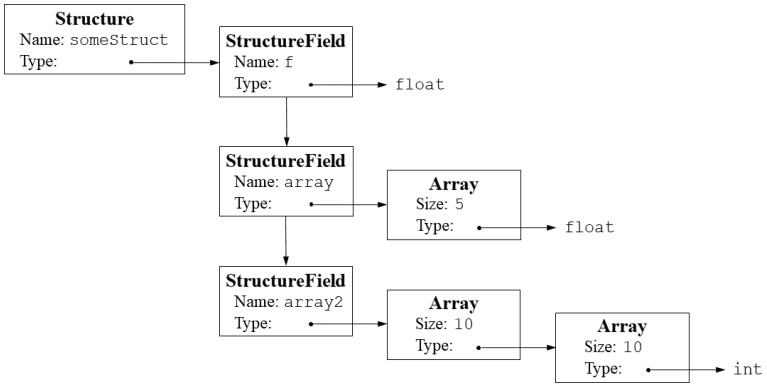

图 2: 类型数据结构

加区分，唯一不同的是，如果是 struct 类型的名字，我们会把表里的 is_struct 置 1。在查找时，如果名字出现冲突，就能直接进行判断。

同时，对于匿名结构体，我们不额外存储它的名字和定义。因为写代码的用户只能通过完整书写结构体的定义，来使用匿名结构体，没有办法直接在使用匿名结构体隐藏的名字。所以无论是名等价还是结构等价，我们都不需要维护匿名结构体的名字。

在保存每一个结构体的时候，我们直接进入 LC，开启下一层的符号表，记录每一个在 LC 和 RC 围起来的区域内出现的符号定义。在碰到 RC，准备退出的时候，我们会把这一层的符号表直接送给这个正在被定义的结构体，然后把相应的指针删除。

4 致谢

感谢刘春旭和张思拓两位同学，课余饭后的讨论，使得在实验过程中累积的疑惑与不解得到逐一解决。