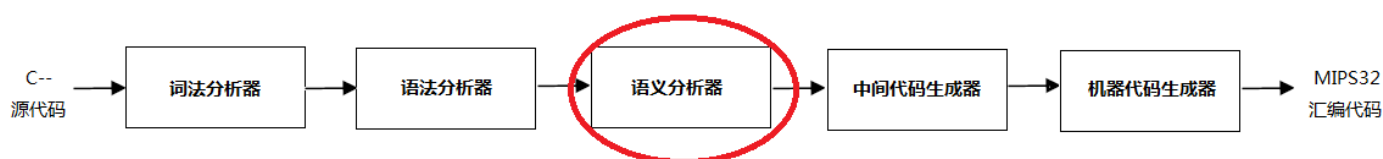


## 编译原理实习 2 语义分析

许畅 陈嘉 朱晓瑞

编译原理课程的实习内容是为一个小型的类 C 语言(C--)实现一个编译器。如果你顺利完成了本课程规定的实习任务，那么不仅你的编程能力将会得到大幅提高，而且你最终会得到一个比较完整的、能将 C--源代码转换成 MIPS 汇编代码的编译器，所得到的汇编代码可以在 SPIM Simulator 上运行。实习总共分为四个阶段：词法和语法分析、语义分析、中间代码生成和目标代码生成。每个阶段的输出是下一个阶段的输入，后一个阶段总是在前一个阶段的基础上完成，如下图所示：




















实习 2 的任务是编写一个程序，该程序读入一个 C--源代码文件，在词法分析和语法分析之后，对其进行语义分析和类型检查（C--语言的词法和语法参见 Grammar.pdf 文件），并打印分析结果。与上一次实习不同的是，这次实习我们已经没有办法依靠任何工具了，所有的检查任务都必须我们要自己手写代码来完成。另外，虽说语义分析在整个编译器的实现中并不是难度最大的任务，但它却是最细致最琐碎的任务，你需要用心地设计符号表和变量类型的数据结构从而在多个方面完善你的编译器中的各种功能。

本次实验中，简单起见我们对 C--语言做如下假设，你可以认为这些就是 C--语言的特性（注意：假设 3、4、5 可能因为你选择的分组任务而有所改变，请务必先阅读实验要求后再动手编程）：

- ◆ 假设 1：整型（int）变量不能与浮点型（float）变量相互赋值或者相互运算
- ◆ 假设 2：仅有 int 型变量才能进行逻辑运算或者作为 if 和 while 语句的条件，也仅有 int 型和 float 型变量能参与算术运算
- ◆ 假设 3：任何一个函数只能进行一次定义，无法进行声明
- ◆ 假设 4：所有变量（包括函数的形参）的作用域都是全局的，即程序中所有变量均不能重名
- ◆ 假设 5：结构体类型间的等价机制采用名等价（name equivalence）的方式
- ◆ 假设 6：函数无法进行嵌套定义
- ◆ 假设 7：结构体中的域不与变量重名，且不同结构体中的域不重名

以上假设 1 至 7 也可视为要求，违反即会导致各种语义错误，不过我们只对后面讨论的 17 种错误类型进行考察。此外，你可以安全地假设输入文件中不包含注释、八进制数、十六进制数、以及指数形式的浮点数，也不包含任何词法或语法错误（除了特别说明的针对选做要求的测试）。

你的程序需要对输入文件进行语义分析（输入文件中可能包含函数、结构体、一维和高维数组）并检查如下错误：

- 变量在使用时未经定义（错误类型 1）
- 函数在调用时未经定义（错误类型 2）
- 变量经过重复定义，或变量与前面定义过的结构体的名字重复（错误类型 3）
- 函数经过重复定义，即同样的函数名出现了不止一次的定义（错误类型 4）
- 赋值号两边表达式类型不匹配（错误类型 5）
- 赋值号左边出现了一个只有右值的表达式（错误类型 6）
- 操作数类型不匹配或操作数类型与操作符不匹配（e. g. 整型变量与数组变量相加减，以及数组/结构体变量与数组/结构体变量相加减）（错误类型 7）
- return 语句返回类型与函数本身的返回类型不匹配（错误类型 8）
- 函数调用时实参与形参的数目或者类型不匹配（错误类型 9）
- 对非数组型变量使用[]（数组访问）操作符（错误类型 10）
- 对普通变量使用()（函数调用）操作符（错误类型 11）
- 数组访问操作符[]中出现非整数（例如 a[1.5]）（错误类型 12）
- 对非结构体型变量使用“.”操作符（错误类型 13）
- 访问结构体中未定义过的域（错误类型 14）
- 结构体中域名的重复定义（指同一结构体中），或者定义时对域进行了初始化（例如 struct A { int a = 0;}就是非法的）（错误类型 15）
- 结构体名与前面定义过的某个结构体或者某个变量的名字重复（错误类型 16）
- 直接使用未定义过的结构体来定义变量（错误类型 17）

以上为每位同学的必做内容。除此之外，你的程序需要完成且**仅能完成**下列三个分组之一中的要求：

#### (1) 分组 2.1

修改对 C 语言假设中的假设 3，使其变为“函数除了在定义之外还可以进行声明”。函数的定义不可以重复出现，但函数声明在相互一致的情况下可以重复出现。任一个函数无论声明与否，其定义必须在源文件中出现。在新的假设 3 下你的程序还需要检查如下错误：

- 函数进行了声明，但没有被定义（错误类型 18）
- 函数的多次声明互相冲突（即函数名一致，但返回类型、形参数量或者形参类型不一致），或者声明与定义之间互相冲突（错误类型 19）

由于 C 语法本身并没有与函数声明相关的产生式，因此你需要先对语法进行适当修改。对于函数声明来说，我们并不要求支持像 `int foo(int, float)` 这样不定义参数名的函数声明。修改的时候要特别留意：你的改动应该以不影响其他错误的检查为原则。

#### (2) 分组 2.2

修改对 C 语言假设中的假设 4，使其变为“变量的定义受可嵌套作用域的影响，外层语句块中

定义的变量名可以在内层语句块中重复定义(但此时在内层语句块中就无法访问到外层语句块的同名变量),内层语句块中定义的变量到了外层语句块中就会消亡,不同函数体内定义的局部变量可以相互重名”。在新的假设 4 下完成错误类型 1 至错误类型 17 的检查。

### (3) 分组 2.3

修改对 C--语言假设中的假设 5,将类型等价机制更改为结构等价(structural equivalence)。例如,虽然名称不同,但两个结构体类型 `struct a{int x; float y;}` 和 `struct b{int y; float z;}` 仍然是等价的类型。注意在结构等价时不要将数组展开来判断,例如 `struct A {int a; struct {float f;int i;} b[10];}` 和 `struct B {struct { int i; float f;} b[10]; int b;}` 就不是等价的。在新的假设 5 下完成错误类型 1 至错误类型 17 的检查。

你可以根据分配到的**任务编号**在任务说明文件(task.pdf)中查找你在整个编译课程实验中**必须**完成的分组。除了必须完成的分组外,完成任何额外的分组都要**倒扣分**!

需要注意的是,由于在后面的实习中还会扩展甚至修改本次实习你已经写好的代码,因此保持一个良好的代码风格、系统地设计代码结构和各模块之间的接口等对于整个实习来讲可以说是相当重要的。

## 输入格式

程序的输入是一个文本文件,其中包含有 C--的源代码,该源代码中可能会有语义错误。

本次实习要求你的程序能够接收一个输入文件名作为参数。例如,若你的程序名为 `cc`、输入文件名为 `test1`、程序和输入文件都位于当前目录下,那么在命令行下运行 `./cc test1` 即可获得以 `test1` 作为输入文件的输出结果。

## 输出格式

本次实习要求你的程序打印到标准输出之上。

对于那些没有语义和类型错误的输入文件,要求你的程序**不输出**任何内容。

对于那些存在语义和类型错误的输入文件,你的程序应当输出相应的错误信息,这些信息包括出错的行号、错误类型以及说明文字,其格式为:

Error type [错误类型] at line [行号]: [说明文字]

说明文字的内容没有具体要求,但是出错的行号和错误类型一定要写对,这是我们判断你输出的错误信息是否正确的唯一标准。请严格遵守实验要求中给定的错误类型分类,否则将严重降低你的分数。

输入文件中可能包含一个或者多个错误(但每行最多只有一个错误),你的程序需要将它们全部检查出来。当然,有些时候输入文件中的一个错误会产生“连锁反应”,导致别的地方出现多个错误(例如,一个未定义的变量在使用时由于无法确定其类型,有可能会同时使包含该变量的表达式产生

一个类型错误)，而我们只会去考察你的程序是否报告了较本质那个的错误（如果难以确定哪一个错误更加本质一些，建议你将所有发现的错误都报告出来）。但是，如果源程序里本来有错而你的程序没有报出来或报告的错误类型不对，又或者源程序里本来没有错但你的程序却凭空报错，都会对你的实习成绩产生负面影响。

## 测试环境

你的程序将在如下环境中被编译并运行：

- ◆ GNU Linux Release: Ubuntu 12.04, kernel version 3.2.0-29
- ◆ GCC version 4.6.3
- ◆ GNU Flex version 2.5.35
- ◆ GNU Bison version 2.5

一般而言，只要避免使用过于冷门的特性，使用其他版本的 Linux 或者 GCC 等，也基本上不会出现兼容性方面的问题。注意，实验检查过程中不会去安装或尝试引用各类方便编程的函数库（如 glib 等），因此请不要在你的程序中使用它们。

## 提交要求

本次实习要求你提交如下内容：

- ◆ Flex、Bison 以及 C 语言的可以被正确编译运行的源代码
- ◆ 一份 PDF 格式的实验报告，内容主要包括：
  - 你的程序实现了哪些功能？简要说明你是如何实现这些功能的。如果因为你的说明不够充分而导致助教没有对你所实现的功能进行测试，那么后果自负。
  - 你所提交上来的程序应当如何编译？不管你使用了脚本也好，准备了 Makefile 也好甚至是单独地逐条命令手工输入进行编译也好，请**详细说明**具体需要键入哪些命令——无法顺利编译将会使你丢失相应分数，并且如果不去找助教进行修正，后面的正确分也会因你的程序无法运行而全部丢失，请谨记这一点。
  - 你的实验报告长度**不得超过 3 页**！因此，你需要好好考虑一下该往实验报告里写些什么。我们的建议是，实验报告中需要你重点描述的应当是你所提交的工作中的亮点，应当是那些你认为**最个性化、最具有独创性**的内容，而那些比较简单的、任何人都可以做出来的内容可以不提或者只简单的提一下，尤其要避免去大段大段地向报告里贴代码。

为了避免大家通过减小字号来变相加长页数限制，我们规定实验报告中所出现的最小字号不得小于五号字（英文 11 号字）。

## 必做内容样例

### 样例输入 1

```
int main()
{
    int i = 0;
    j = i + 1;
}
```

### 样例输出 1

样例程序中变量“j”未定义，因此你的程序应输出如下错误提示信息：

Error type 1 at line 4: Undefined variable “j”

### 样例输入 2

```
int main()
{
    int i = 0;
    inc(i);
}
```

### 样例输出 2

样例程序中函数“inc”未定义，因此你的程序应输出如下错误提示信息：

Error type 2 at line 4: Undefined function “inc”

### 样例输入 3

```
int main()
{
    int i, j;
    int i;
}
```

### 样例输出 3

样例程序中变量“i”被重复定义，因此你的程序应输出如下错误提示信息：

Error type 3 at line 4: Redefined variable "i"

### 样例输入 4

```
int func(int i)
{
    return i;
}

int func()
{
    return 0;
}

int main()
{
}
```

### 样例输出 4

样例程序中函数“func”重复定义，因此你的程序应输出如下错误提示信息：

Error type 4 at line 6: Redefined function "func"

## 样例输入 5

```
int main()
{
    int i;
    i = 3.7;
}
```

## 样例输出 5

样例程序中错将一个浮点数赋值给一个整型变量，因此你的程序应输出如下错误提示信息：

Error type 5 at line 4: Type mismatched

## 样例输入 6

```
int main()
{
    int i;
    10 = i;
}
```

## 样例输出 6

样例程序中整数“10”出现在赋值号的左边，因此你的程序应输出如下错误提示信息：

Error type 6 at line 4: The left-hand side of an assignment must be a variable

## 样例输入 7

```
int main()
{
    float j;
    10 + j;
}
```

## 样例输出 7

样例程序中表达式“10+j”的两个操作数的类型不匹配，因此你的程序应输出如下错误提示信息：

Error type 7 at line 4: Operands type mismatched

## 样例输入 8

```
int main()
{
    float j = 1.7;
    return j;
}
```

## 样例输出 8

样例程序中“main”函数返回值的类型不正确，因此你的程序应输出如下错误提示信息：

Error type 8 at line 4: The return type mismatched



## 样例输入 9

```
int func(int i)
{
    return i;
}

int main()
{
    func(1, 2);
}
```

## 样例输出 9

样例程序中调用函数“func”时实参数目不正确，因此你的程序应输出如下错误提示信息：

Error type 9 at line 8: The method "func(int)" is not applicable for the arguments "(int, int)"

## 样例输入 10

```
int main()
{
    int i;
    i[0];
}
```

## 样例输出 10

样例程序中变量“i”非数组型变量，因此你的程序应输出如下错误提示信息：

Error type 10 at line 4: "i" must be an array

## 样例输入 11

```
int main()
{
    int i;
    i(10);
}
```

## 样例输出 11

样例程序中变量“i”不是函数，因此你的程序应输出如下错误提示信息：

Error type 11 at line 4: “i” must be a function

## 样例输入 12

```
int main()
{
    int i[10];
    i[1.5] = 10;
}
```

## 样例输出 12

样例程序中数组访问符中出现非整型数“1.5”，因此你的程序应输出如下错误提示信息：

Error type 12 at line 4: Operands type mistaken

### 样例输入 13

```
struct Position
{
    float x, y;
};

int main()
{
    int i;
    i.x;
}
```

### 样例输出 13

样例程序中变量“i”非结构体类型变量，因此你的程序应输出如下错误提示信息：

Error type 13 at line 9: Illegal use of "."

### 样例输入 14

```
struct Position
{
    float x, y;
};

int main()
{
    struct Position p;
    if (p.n == 3.7)
        return 0;
}
```

### 样例输出 14

样例程序中结构体变量“p”访问了未定义的域“n”，因此你的程序应输出如下错误提示信息：

Error type 14 at line 9: Un-existed field "n"

## 样例输入 15

```
struct Position
{
    float x, y;
    int x;
};

int main()
{
}
```

## 样例输出 15

样例程序中结构体的域“x”被重复定义，因此你的程序应输出如下错误信息：

Error type 15 at line 4: Redefined field 'x'

## 样例输入 16

```
struct Position
{
    float x;
};

struct Position
{
    int y;
};

int main()
{
}
```

## 样例输出 16

样例程序中两个结构体的名重复，因此你的程序应输出如下错误信息：

Error type 16 at line 6: Duplicated name 'Position'

## 样例输入 17

```
int main()
{
    struct Position pos;
}
```

## 样例输出 17

样例程序中结构体“Position”未定义，因此你的程序应输出如下错误信息：

```
Error type 17 at line 3: Undefined struct 'Position'
```

## 分组内容样例

### 样例输入 1

```
int func(int a);

int func(int a)
{
    return 1;
}

int main()
{
}
```

### 样例输出 1

对于**需要**完成分组 2.1 的同学，这个样例程序不存在词法、语法及语义错误，因此没有输出。

对于**不需要**完成分组 2.1 的同学，这个样例程序不能通过你的程序的语法分析部分。你的程序**应该**给出输入文件包含**语法错误**的提示信息：

```
Error type B at line 1: Incomplete definition of
function "func"
```

## 样例输入 2

```
struct Position
{
    float x,y;
};

int func(int a);

int func(struct Position p);

int main()
{
}
```

## 样例输出 2

对于**需要**完成分组 2.1 的同学，这个样例程序包含两个语义错误：（1）函数“func”的两次声明不一致；（2）函数“func”未定义。因此你的程序应输出如下错误提示信息：

```
Error type 19 at line 8: Inconsistent declaration of
function "func"
Error type 18 at line 6: Undefined function "func"
```

注意我们对错误信息的顺序不做要求。

对于**不需要**完成分组 2.1 的同学，这个样例程序不能通过你的程序的语法分析部分。你的程序**应该**给出输入文件包含**语法错误**的提示信息：

```
Error type B at line 6: Incomplete definition of
function "func"
Error type B at line 8: Incomplete definition of
function "func"
```

### 样例输入 3

```
int func()
{
    int i = 10;
    return i;
}

int main()
{
    int i;
    i = func();
}
```

### 样例输出 3

对于**需要**完成分组 2.2 的同学，这个样例程序不存在词法、语法及语义错误，因此没有输出。

对于**不需要**完成分组 2.2 的同学，样例程序中变量“i”被重复定义，因此你的程序应输出如下错误提示信息：

Error type 3 at line 9: Redefined variable "i"

### 样例输入 4

```
int func()
{
    int i = 10;
    return i;
}

int main()
{
    int i;
    int i,j;
    i = func();
}
```

## 样例输出 4

对于**需要**完成分组 2.2 的同学，“main”函数中变量“i”重复定义，因此你的程序应输出如下错误提示信息：

Error type 3 at line 10: Redefined variable "i"

对于**不需要**完成分组 2.2 的同学，样例程序中变量“i”被重复定义两次，因此你的程序应输出如下错误提示信息：

Error type 3 at line 9: Redefined variable "i"  
Error type 3 at line 10: Redefined variable "i"

## 样例输入 5

```
struct Temp1
{
    int i;
    float j;
};

struct Temp2
{
    int x;
    float y;
};

int main()
{
    struct Temp1 t1;
    struct Temp2 t2;
    t1 = t2;
}
```

## 样例输出 5

对于**需要**完成分组 2.3 的同学，这个样例程序不存在词法、语法及语义错误，因此没有输出。

对于**不需要**完成分组 2.3 的同学，样例程序中语句“t1 = t2;”赋值号两边变量的类型不匹配，因此你的程序应输出如下错误提示信息：

Error type 5 at line 17: Type mismatched



## 样例输入 6

```
struct Temp1
{
    int i;
    float j;
};

struct Temp2
{
    int x;
};

int main()
{
    struct Temp1 t1;
    struct Temp2 t2;
    t1 = t2;
}
```

## 样例输出 6

对于**需要**完成分组 2.3 的同学，样例程序中语句“t1 = t2;”赋值号两边变量的类型不匹配，因此你的程序应输出如下错误提示信息：

Error type 5 at line 16: Type mismatched

对于**不需要**完成分组 2.3 的同学，你的程序应输出与上述提示信息相同的结果。