实验5 语义分析

实验5的任务是在词法和语法分析的基础上，编写一个程序，对使用SysY语言书写的源代码进行静态语义检查，并打印分析结果。实验的实现方式可以手工编写程序，也可以探索在Bison中增加语义检查规则来实现。

需要注意的是，由于在后面的实验中还会用到本次实验已完成的代码，因此，建议保持良好的代码风格，系统地设计代码结构和各模块之间的接口。

5.1 实验要求

程序要能够查出SysY源代码中可能包含的语义错误。本次实验中，SysY语言的语义特性符合SysY语言定义(2022版)中的语义约束描述，我们简单归纳如下：

1. 特性1：一个SysY程序由单个文件构成，文件内容对应语法单元为CompUnit，其中必须存在一个标识符为“main”、无参数、返回类型为int的FuncDef（函数定义）
2. 特性2：每个CompUnit对应的顶层变量或常量声明语句、函数定义，都不能重复定义同名标识符，即使类型不同也不行。【简单的说，就是不允许在同个作用域下具有同名的标识符（包括变量、常量、函数名）】
3. 特性3：任何函数只进行一次定义，无法进行函数声明【同学们可以关注一下SysY语言定义中的FuncDef】
4. 特性4：与C语言标准一致，函数无法进行嵌套定义，即函数内部不能再定义函数
5. 特性5：作为if和while条件部分的语法单位Cond是一个LOrExp，需要特别注意逻辑运算表达式，单目运算符！只能出现在LOrExp，而不能在Exp中出现。
6. 特性6：函数调用时实际参数和函数定义中的参数个数和类型必须保持完全匹配。
7. 特性7：SysY允许类型隐式转换，本次实验我们假设int和float不能互相赋值或互相运算。

各小组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、一维或多维数组），并检查如下类型的错误：

1. **错误类型1**：变量未声明
2. **错误类型2**：变量重复声明
3. **错误类型3**：函数在调用时未定义
4. **错误类型4**：函数重复定义（同样的函数名出现了不止一次定义）
5. **错误类型5**：把变量当做函数调用，如对普通变量使用括号(...)或()运算符（当函数调用），
6. **错误类型6**：对函数名的不当引用（如把函数名当做普通变量来引用）
7. **错误类型7**：对数组的不当引用，如数组访问运算符"[...]"中出现非整数表达式，即数组变量的下标不是整型
8. **错误类型8**：对非数组变量使用数组访问"[...]"运算符
9. **错误类型9**：函数调用时参数个数或类型不匹配
10. **错误类型10**：return语句返回的类型与函数定义的返回类型不匹配
11. **错误类型11**：操作数类型不匹配，或操作数类型与操作符不匹配，如：整型变量与数组变量相加减，或数组变量与数组变量相加减。
12. **错误类型12**：break语句不在循环体内
13. **错误类型13**：continue语句不在循环体内

    注意：语法分析能发现的错误不包含在上面

要求至少能识别上述3个以上的错误类型，程序在输出错误提示信息时，需要输出具体的错误类型、出错的位置（源程序的行号）以及相关的说明文字。

5.2 输入格式

程序输入是一个包含SysY源代码的文本文件，程序需要能够接受一个输入文件名作为参数。例如：假设程序名为ss，输入文件名为test1.sy，程序和输入文件都在当前目录下，那么在Linux命令行下运行 ./ss test1.sy即可获得test1.sy作为输入文件的输出结果。

5.3 输出格式

实验要求通过标准输出打印程序的运行结果（也可以单独输出到文件，格式保持一致）。对于没有语义错误的输入文件，程序不需要输出任何内容，也可以打印“success"字样。对于那些包含语义错误的输入文件，程序应该输出相应的错误信息，包括错误类型、出错的行号以及说明文字，格式如下：

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

说明文字的内容不做具体要求，但是错误类型和行号要正确。**假设输入文件中可能包含一个或者多个错误，但同一行最多只有一个错误**。每一条错误信息在输出中单独占一行。

5.4 提交要求

实验5要求提交如下内容：

1. 可被正确编译执行的源程序（使用工具的话，需要提供工具源码Flex/Bison和修改后的C代码，其他语言实现则需要说明配置环境）
2. 一份实验报告的PDF文件，内容包括：
   - [ ] 程序实现的主要功能，简要说明怎么实现的。
   - [ ] 程序如何进行编译？ 特别是采用冷门语言编写的代码
   - [ ] 实验报告总长度不要超过4页。重点描述的是自己程序的亮点和不一样的地方，对于相对简单的内容可以不提或简单提一下，杜绝大段粘贴源码。实验报告字体最小字号是五号字。

5.5 样例

例5.1 输入(行号是标识需要，并非样例输入的一部分，下同)

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

输出: 该程序存在语义错误。第4行中的变量j没有被声明过，因此程序需要输出以下错误信息

```
Error type 1 at Line 4: undefined variable "j".
```

例5.2 输入

```
int main()
{
  int i = 1;
  inc(i);
  return 0;
}
```

输出: 该程序存在语义错误。第4行中的函数调用inc(i)，inc没有在程序中被定义，因此程序需要输出以下错误信息

```
Error type 3 at Line 4: undefined function "inc".
```

例5.3 输入

```
int main()
{
  int i, j;
  int i;
  return 0;
}
```

输出: 该程序存在语义错误。第4行中的变量i在同一个作用域中被重复定义，因此程序需要输出以下错误信息

```
Error type 2 at Line 4: redefined variable "i".
```

例5.4 输入

```
int main()
{
  float i;
  i+10;
  return 0;
}
```

输出: 该程序存在语义错误。第4行中的表达式i+10，其中加法两个操作数类型不一致，因此程序需要输出以下错误信息

```
Error type 11 at Line 4: type mismatched for operands.
```

例5.5 输入

```
int main()
{
  int a[10];
  a[1.5]=10;
  return 0;
}
```

输出: 该程序存在语义错误。第4行中的表达式a[1.5]，数组引用出现了非整型数据，因此程序需要输出以下错误信息

```
Error type 7 at Line 4: "1.5" is not an integer.
```

例5.6 输入

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

输出: 该程序存在语义错误。第4行中的return语句返回类型与函数定义返回类型不一致，因此程序需要输出以下错误信息

```
Error type 10 at Line 4: type mismatched for return.
```

例5.7 输入

```
int func(int i)
{
  return i;
}
int main()
{
  func(2, 3)
  return 0;
}
```

输出: 该程序存在语义错误。第7行中的函数调用func(2,3)，参数个数与函数定义不一致，，因此程序需要输出以下错误信息

```
Error type 9 at Line 7: function "func" only need one argument.
```