第一部分：词法分析

1. 内容简介：

在第一部分，我们要构建一个词法分析器来识别 Pony 语言中的各种词法单元 (Token)，包括关键字（如 var、def 和 return）、特殊符号、数字以及变量名/函数名等。我们要通过相关函数来获取 Token，判断其合法性，并针对非法格式输出错误信息。

2. 功能实现：

Tips：

- 在Lexer.h搜索"TODO"，可以看到需要实现的相关函数以及具体要求。
- 在处理非法情况时，要求编译器在终端输出尽可能详细的错误信息。
- 在实现具体功能之前，须阅读Lexer.h中已有代码。

2.1 词法分析功能实现：

文件地址： /pony_compiler/pony/include/pony/Lexer.h

要求实现以下功能：

1). 实现成员函数getNextChar()。

① 注意几种corner case的特殊处理，比如读到某行结尾，读到文档结尾等情况。

② 注意行列等位置信息的更新。

2). 补充成员函数getTok()。

① 能够识别“return”、“def”和“var”三个关键字

② 能够识别标识符

- 标识符由字母、数字或下划线组成
- 标识符以字母或下划线开头
- 标识符中有数字时，数字不可连续出现，例如：有效的标识符可以是 ab3c, _b, placeholder 等。

③ 改进识别number的方法，使编译器可以识别并在终端报告非法number。合法number表示为 $[0-9]^*.[0-9]^+$ ，此外均为非法表示，例如：9.9.9，9..9，..9，9..等。

2.2 词法分析验证程序实现：

文件地址： /pony_compiler/pony/ponyc.cpp + /pony_compiler/pony/include/pony/Lexer.h

要求实现以下功能：

补充ponyc.cpp文件中“词法分析器正确性”验证程序int dumpToken()。扩展Lexer.h文件中getTok()函数，在识别每种Token的同时，将其存放在某种数据结构中，以便最终在终端输出。

① 输入文件为Pony语言定义的函数等。

② 如果词法分析器没有识别出错误，则按顺序输出识别到的Token。

③ 如果词法分析器识别出错误，则在终端输出详细的错误信息
④ 输出信息的具体形式可参考接下来第3节中的测试示例。

3.实验验证：

在对词法分析器构建完毕后，可以通过运行测试用例test_1至test_7来检查词法分析器的正确性。以test_1为例：build pony并执行下面测试用例，以验证词法分析器能够识别出各种合法的词法单元：

```
$ cmake --build . --target pony
$ ./bin/pony ../../test/test_1.pony -emit=token
```

如果执行结果如下图所示，表示词法分析器分析正确。

```
root@e062ec9d43ba:/home/workspace/pony_compiler/src/build# ./bin/pony ../../test/test_1.pony -emit=token
def main ( ) { var a [ 2 ] [ 3 ] = [ 1 , 2 , 3 , 4 , 5 , 6 ] ; } EOF
```

以test_3为例：build pony并执行下面测试用例，以验证词法分析器能够识别出各种非法的词法单元并输出错误信息：

```
$ cmake --build . --target pony
$ ./bin/pony ../../test_3.pony -emit=token
```

如果执行结果如下图所示，表示词法分析器正确识别了非法的标识符。

```
root@e062ec9d43ba:/home/workspace/pony_compiler/src/build# ./bin/pony ../../test/test_3.pony -emit=token
def main ( ) { var 5:12: ERROR: Continuous digits in identifier. [ 2 ] [ 3 ] = [ 1 , 2 , 3 , 4 , 5 , 6 ] ; } EOF
```