词法分析实验报告

胡基宸 521021910143

1. 实验环境

虚拟机软件：Vmware

操作系统：Ubuntu 20.04

依赖工具链：git, cmake, lld, clang, ninja

2. 思路与方法

在 lexer 中定义一个名为 tokens 的 vector<string>作为数据结构存储所有识别到的 token，并存储一个 bool 类型的变量 is_error 来记录识别过程中是否发生了异常，在 ponyc.cpp 中如果无异常，就将 tokens 遍历输出，如果有异常就不输出，在发生异常时会有错误信息的终端提示。

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。(curLineNum,curCol)

```
int getNextChar() {
    /*
     *
     *  Write your code here.
     *
     */
    if(curLineBuffer.empty()) return EOF;
    ++curCol;
    auto nextchar = curLineBuffer.front();
    curLineBuffer = curLineBuffer.drop_front();
    if (curLineBuffer.empty())
      curLineBuffer = readNextLine();
    if (nextchar == '\n') {
      ++curLineNum;
      curCol = 0;
    }
    return nextchar;
  }
```

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：

- 标识符以字母或下划线开头
- 标识符由字母、数字和下划线组成
- 按照使用习惯，要求标识符中有数字时，数字须位于标识符末尾

例如：有效的标识符可以是 a123, b_4, 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，非法表示包括：9.9.9， 9..9，.123 等。

识别标识符：

```
if (isalpha(lastChar) || lastChar == '_') {//indicator or key word
      std::string varStr;
      bool no_digit = true;
      bool error_indicator = false;

      while (isalnum(lastChar) || lastChar == '_'){
        if (isdigit(lastChar)) {
          no_digit = false;
        }
        if (!no_digit) {
          if (isalpha(lastChar) || lastChar == '_') {
            error_indicator = true;//the digit is not at the end
            is_error = true;
            cout << "Token error (" << curLineNum << ", " << curCol
            << "): digits should be at the end of an indicator" << endl;
            break;
          }
        }
        varStr += lastChar;
        lastChar = Token(getNextChar());
      }

      if (!error_indicator) {
        if (varStr == "return") {tokens.push_back("return"); return
tok_return;}
        if (varStr == "def") {tokens.push_back("def"); return tok_def;}
        if (varStr == "var") {tokens.push_back("var"); return tok_var;}
        identifierStr = varStr;
        tokens.push_back(identifierStr);
        return tok_identifier;
      }
    }
```

识别数字：

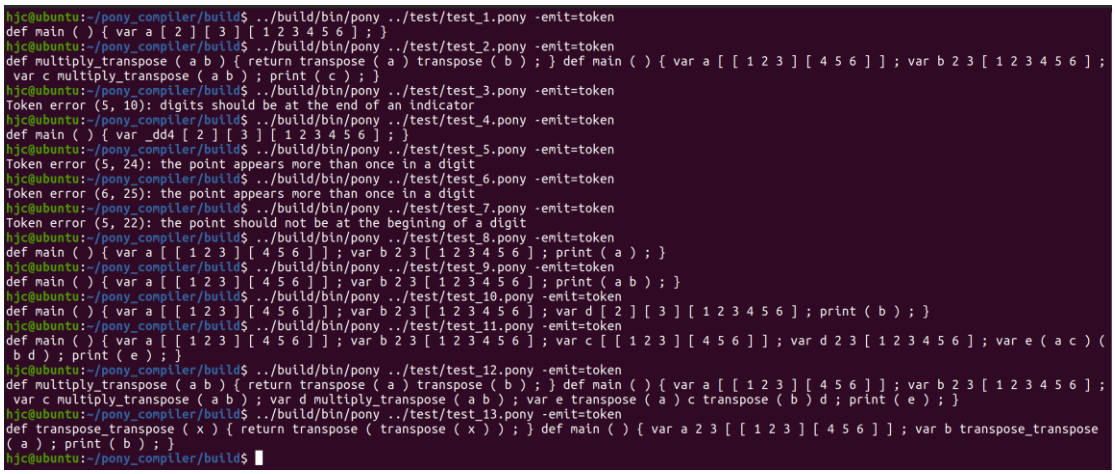
```
//TODO: 3. 改进识别数字的方法，使编译器可以识别并在终端报告非法数字，非法表示包括：9.
9.9, 9..9, .123等。
    bool first_digit = true;
    bool error_digit = false;
    int point_count = 1;
    if (isdigit(lastChar) || lastChar == '.') {//digit
      std::string numStr;
      do {
        //'.' at the begining
        if(first_digit && lastChar == '.'){
          error_digit = true;
          is_error = true;
          cout << "Token error (" << curLineNum << ", " << curCol
            << "): the point should not be at the begining of a digit"
<< endl;
        }
        first_digit = false;

        if(lastChar == '.'){
          point_count--;S
          if(point_count < 0){
            error_digit = true;//point appears more than once
            is_error = true;
            cout << "Token error (" << curLineNum << ", " << curCol
            << "): the point appears more than once in a digit" << endl;
          }
        }
        numStr += lastChar;
        lastChar = Token(getNextChar());
      } while (!error_digit && (isdigit(lastChar) || lastChar == '.'));

      numVal = strtod(numStr.c_str(), nullptr);
      if(!error_digit) tokens.push_back(numStr);
      return tok_number;
    }
```

3）补充 ponyc.cpp 文件中“词法分析器正确性”验证程序 int dumpToken()。

```
while(lexer.lastChar != EOF){
    lexer.getNextToken();
  }
  if(!lexer.is_error){
    for(auto x: lexer.tokens){
      cout << x << ' ';
    }
    cout << endl;
  }
```

3. 实验结果与分析

全部 13 个验证结果如下：



可以看出，当发生异常时会有对应的终端提示，并注明对应的行列号，没有异常时正常输出识别到的结果，并用空格分开显示。

4. 遇到的问题与解决方法

（1） 实验环境是 ubuntu20.04，一开始用 ubuntu22.04 造成了一些工具链版本问题。

（2） 列号与实验实例有细小差异，因为我的终端提示列号用的是刚好发生错误的确切地方，而不是异常 token 的开始位置。