

**计算机学院本科生实验报告**

实验课程：编译原理

实验项目： C++单词拼接器

指导教师： 黄煜廉

开课时间： 2021年9月6日

班级： 计算机科学与技术3班

1. 软件名称

C++单词拼接器；

1. 设计目的

模拟C++代码底层的编译过程，相当于设计一个等价与C++词法分析器功能的软件，加深对C++词法分析器的学习理解；

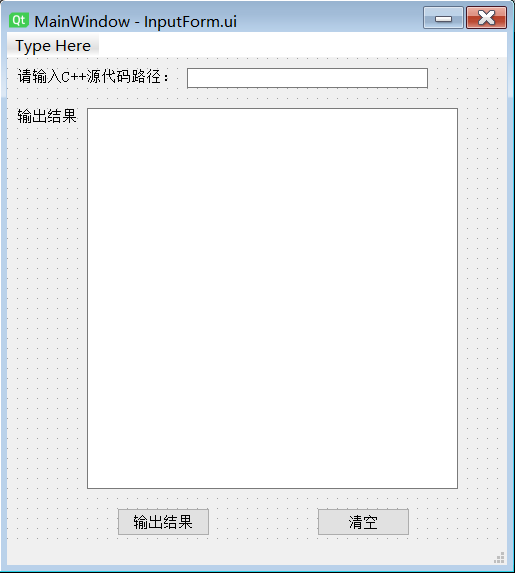
1. 开发环境

本软件在Windows10操作系统下使用PyQt5设计实现，python版本号为3.9.1，软件UI使用QT Designer设计完成；

1. 软件功能

把C++源代码中的各类单词（记号）进行拼装分类。C++语言包含了几种类型的单词（记号）：标识符，关键字，数（包括整数、浮点数），字符串、注释、特殊符号（分界符）和运算符号等；打开一个C++源文件，列出所有可以拼装的单词（记号）；

1. 功能设计
2. 软件UI设计，要求要有输入目标文件路径的输入框以及显示输入结果的界面；



在终端执行代码：pyuic5 -o InputForm.py InputForm.ui后，的得到下面代码：

|  |
| --- |
| # -\*- coding: utf-8 -\*- # Form implementation generated from reading ui file 'InputForm.ui' # # Created by: PyQt5 UI code generator 5.6 # # WARNING! All changes made in this file will be lost! from PyQt5 import QtCore, QtGui, QtWidgets from refer import \* class Ui\_MainWindow(object):  def setupUi(self, MainWindow):  MainWindow.setObjectName("MainWindow")  MainWindow.resize(500, 533)  self.centralwidget = QtWidgets.QWidget(MainWindow)  self.centralwidget.setObjectName("centralwidget")  self.label = QtWidgets.QLabel(self.centralwidget)  self.label.setGeometry(QtCore.QRect(10, 10, 151, 16))  self.label.setObjectName("label")  self.pushButton = QtWidgets.QPushButton(self.centralwidget)  self.pushButton.setGeometry(QtCore.QRect(110, 450, 93, 28))  self.pushButton.setObjectName("pushButton")  self.pushButton\_2 = QtWidgets.QPushButton(self.centralwidget)  self.pushButton\_2.setGeometry(QtCore.QRect(310, 450, 93, 28))  self.pushButton\_2.setObjectName("pushButton\_2")  self.textEdit\_2 = QtWidgets.QTextEdit(self.centralwidget)  self.textEdit\_2.setGeometry(QtCore.QRect(80, 50, 371, 381))  self.textEdit\_2.setReadOnly(True)  self.textEdit\_2.setObjectName("textEdit\_2")  self.label\_2 = QtWidgets.QLabel(self.centralwidget)  self.label\_2.setGeometry(QtCore.QRect(10, 50, 61, 16))  self.label\_2.setObjectName("label\_2")  self.lineEdit = QtWidgets.QLineEdit(self.centralwidget)  self.lineEdit.setGeometry(QtCore.QRect(180, 10, 241, 20))  self.lineEdit.setObjectName("lineEdit")  MainWindow.setCentralWidget(self.centralwidget)  self.menubar = QtWidgets.QMenuBar(MainWindow)  self.menubar.setGeometry(QtCore.QRect(0, 0, 500, 23))  self.menubar.setObjectName("menubar")  MainWindow.setMenuBar(self.menubar)  self.statusbar = QtWidgets.QStatusBar(MainWindow)  self.statusbar.setObjectName("statusbar")  MainWindow.setStatusBar(self.statusbar)   self.retranslateUi(MainWindow)  QtCore.QMetaObject.connectSlotsByName(MainWindow)   def retranslateUi(self, MainWindow):  \_translate = QtCore.QCoreApplication.translate  MainWindow.setWindowTitle(\_translate("MainWindow", "MainWindow"))  self.label.setText(\_translate("MainWindow", "请输入C++源代码路径："))  self.pushButton.setText(\_translate("MainWindow", "输出结果"))  self.pushButton\_2.setText(\_translate("MainWindow", "清空"))  self.label\_2.setText(\_translate("MainWindow", "输出结果：")) |

1. 读入文件后，对代码进行第一次扫描，首先要先删除代码的注释，再将逐行代码以字符串的类型存入到list中；

|  |
| --- |
| ComPlier = Complier() #自定义的类，用于做词法分析 SourceProgram = [] #Filepath = input("请输入文件路径：") Filepath = input("请输入文件路径：") for line in open(Filepath, 'r', encoding='UTF-8-sig'):  line = line.replace('\n', '')  SourceProgram.append(line)  SourceProgram = ComPlier.DeleteNote(SourceProgram) #去注释 |

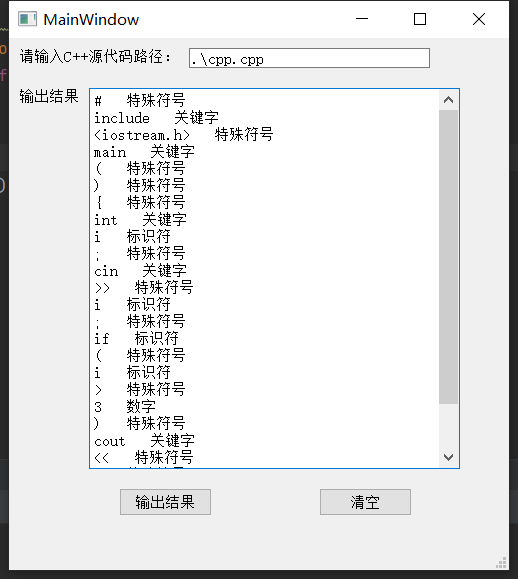
1. 再扫描第2步的list，根据代码中的空格与回车等分隔符对代码做初步的分词，并将分好的单词再存入到一个list中；

|  |
| --- |
| SourceProgram = ComPlier.RemoveSpace(SourceProgram) #去空格 SourceProgram = ComPlier.Reader(SourceProgram) #初步分割 |

1. 再扫描第3步的list，增加词法上的约束，进一步对单词进行分割或者合并；

|  |
| --- |
| SourceProgram = ComPlier.combine\_head(SourceProgram) #进一步分割 ComPlier.JugeMent(SourceProgram) #输出分析结果 |

1. 显示输出结果；



1. 总结

最初设计C++词法拼接器，没有去归纳总结词法拼接器的规律或者是C++单词拼接的规则，甚至是判断字符的顺序不对，整个代码只扫描一遍，导致程序写的很乱，在学习了有穷自动机之后，再结合网上的一些设计步骤，开始分步骤设计，多遍扫描源代码，才得以实现预期的功能。