学号 姓名

学校 地址

实验一

词法分析

目录

[**1.实验目的** 2](#_Toc130907619)

[**2.实验要求** 2](#_Toc130907620)

[**3.实验步骤** 2](#_Toc130907621)

[3.1程序功能 2](#_Toc130907622)

[3.2程序实现方式 2](#_Toc130907623)

[3.3开发环境和框架 2](#_Toc130907624)

[3.4运行环境 3](#_Toc130907625)

[3.5设计方案 3](#_Toc130907626)

[**4.程序实现** 3](#_Toc130907627)

[4.1代码组织 3](#_Toc130907628)

[4.2实体层 4](#_Toc130907629)

[4.2.1数据实体 4](#_Toc130907630)

[4.2.2数据标识对象 6](#_Toc130907631)

[4.3业务层 7](#_Toc130907632)

[4.3.1读取关键字表和运算符表 7](#_Toc130907633)

[4.3.2生成变量唯一ID 9](#_Toc130907634)

[4.3.3格式化词法分析日志 9](#_Toc130907635)

[4.3.4异常处理 11](#_Toc130907636)

[4.3.5业务层接口和实现 11](#_Toc130907637)

[4.4界面层 17](#_Toc130907638)

[4.4.1主界面 18](#_Toc130907639)

[4.4.2符号表显示控件 20](#_Toc130907640)

[4.2.3结果展示界面 21](#_Toc130907641)

[4.5工程配置文件 28](#_Toc130907642)

[**5.运行效果** 29](#_Toc130907643)

**1.实验目的**

设计模拟词法分析程序

**2.实验要求**

设计一个含简单变量的词法分析器。词法的格式要求：

变量名只包含字母数字，且开头必须是字母。

变量类型只有四种：integer，real，bool，char

关键字和运算符已经给出。

变量的值只能是数字

用分号代表代码换行

输入：一段要分析的字符串，可能有多行

输出：变量表（符号表），包括变量名称，id，类型

词法分析日志：每一个符号是什么类型逐一输出分析结果

**3.实验步骤**

3.1程序功能

检查变量名，变量类型，打印符号表

检查程序的词法是否正确，有没有变量未说明就使用的情况，给出每个符号的分析结果

具有图形用户界面，以交互形式输入源程序，并以图形方式展示结果

3.2程序实现方式

Java

3.3开发环境和框架

JDK1.8，Android framework SDK API 32

3.4运行环境

Windows Subsystem for Android 2302.40000.8.0 在 Windows 10X build22624.1470上

3.5设计方案

传统的扫描需要多个指针来回移动，效率很高但是实现非常麻烦。这里借助一些现代字符串API（spilt，trim，replace，substring等）（本质上原理一样，但是可以简化代码），用根据关键字分割的方法，从一长串字符分成若干行，每一行再分成若干单词。实现“大事化小，小事化了”的效果。这样做编程相对简单一些。

对每一个单词运行相应的判断逻辑，立马可以分析出是否正确。并打印相关的日志。实时给出分析结果

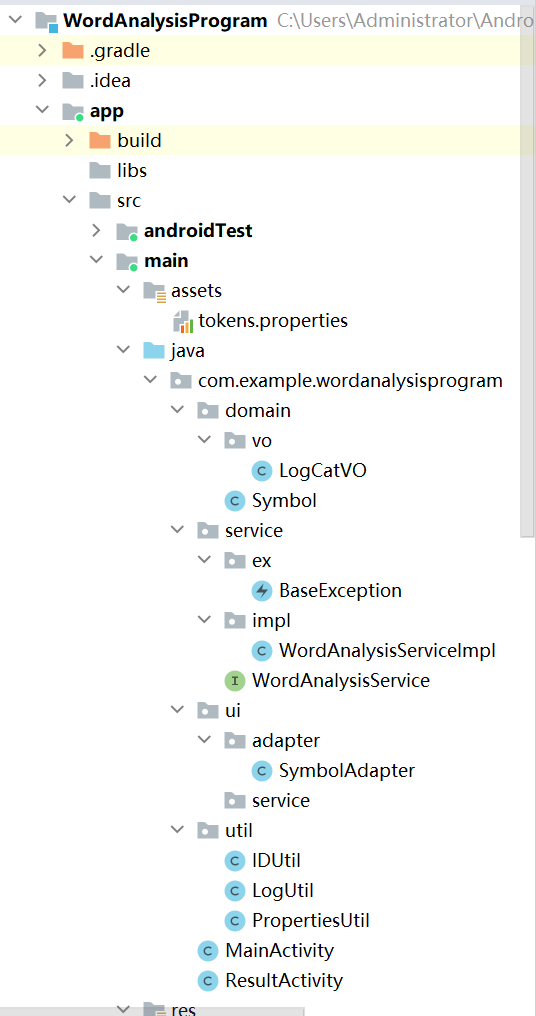
对于出现的错误要及时抛出异常，中止分析

对于代码实现部分，检查变量是否以定义，如果没有，也要报错。

**4.程序实现**

4.1代码组织

类结构：



4.2实体层

4.2.1数据实体

符号表项实体，用一个pojo代表。重写equals，变量名想等就认为是同一个变量

|  |
| --- |
| domain/Symbol.java |
| package com.example.wordanalysisprogram.domain;  import java.util.ArrayList; import java.util.List; import java.util.Objects;  public class Symbol {   public static final List<String> *types*=new ArrayList<>();  static {  *types*.add("integer");  *types*.add("real");  *types*.add("bool");  }   private String id;  private String name;  private String type;   public String getId() {  return id;  }   public void setId(String id) {  this.id = id;  }   public String getName() {  return name;  }   public void setName(String name) {  this.name = name;  }   public String getType() {  return type;  }   public void setType(String type) {  this.type = type;  }   @Override  public boolean equals(Object o) {  if (this == o) return true;  if (o == null || getClass() != o.getClass()) return false;  Symbol vo = (Symbol) o;  return Objects.*equals*(name, vo.name);  }   @Override  public int hashCode() {  return Objects.*hash*(name);  }   @Override  public String toString() {  return "SymbolVO{" +  "id='" + id + '\'' +  ", name='" + name + '\'' +  ", type='" + type + '\'' +  '}';  } } |

4.2.2数据标识对象

业务层返回对象，包含类型（变量表还是日志），如果是变量表则包含变量表list，否则包含日志list

|  |
| --- |
| domain/LogCatVO.java |
| package com.example.wordanalysisprogram.domain.vo;  import com.example.wordanalysisprogram.domain.Symbol;  import java.util.List;  public class LogCatVO {   public static final int *VARIABLE\_ANALYSIS*=0;  public static final int *CODE\_ANALYSIS*=1;   private int type;  private boolean flag;  private List<Symbol> symbols;  private List<String> logs;   public int getType() {  return type;  }   public void setType(int type) {  this.type = type;  }   public boolean isFlag() {  return flag;  }   public void setFlag(boolean flag) {  this.flag = flag;  }   public List<Symbol> getSymbols() {  return symbols;  }   public void setSymbols(List<Symbol> symbols) {  this.symbols = symbols;  }   public List<String> getLogs() {  return logs;  }   public void setLogs(List<String> logs) {  this.logs = logs;  } } |

4.3业务层

4.3.1读取关键字表和运算符表

配置文件位于项目类路径下assets文件夹

|  |
| --- |
| util/PropertiesUtil.java |
| package com.example.wordanalysisprogram.util;  import android.content.Context;  import java.io.IOException; import java.io.InputStream; import java.util.ArrayList; import java.util.Collections; import java.util.List; import java.util.Properties;  public class PropertiesUtil {  public static List<String> getOperators(Context context)  {  Properties props = new Properties();  try {  InputStream in = context.getAssets().open("tokens.properties");  props.load(in);   String ops=props.getProperty("operators");  String[]o= ops.split(",");  List<String> res=new ArrayList<>();  Collections.*addAll*(res, o);  return res;  } catch (IOException e) {  e.printStackTrace();  }  return null;  }   public static List<String> getReservedTokens(Context context)  {  Properties props = new Properties();  try {  InputStream in = context.getAssets().open("tokens.properties");  props.load(in);   String ops=props.getProperty("reserved\_tokens");  String[]o= ops.split(",");  List<String> res=new ArrayList<>();  Collections.*addAll*(res, o);  return res;  } catch (IOException e) {  e.printStackTrace();  }  return null;  } } |

下面是配置文件

|  |
| --- |
| CLASS\_PATH/assets/tokens.properties |
| operators=+,-,\*,/,%,(,),+=,-=,\*=,/=,<,>,=,<=,>=,==,!=,[,],:= reserved\_tokens=begin,end,if,while,for,switch,else,then |

4.3.2生成变量唯一ID

|  |
| --- |
| util/IDUtil.java |
| package com.example.wordanalysisprogram.util;  import java.util.UUID;  public class IDUtil {  public static String getUUID()  {  return UUID.*randomUUID*().toString();  } } |

4.3.3格式化词法分析日志

生成可打印字符串，包含行号，符号名称，分析结果，返回给UI展示

|  |
| --- |
| util/LogUtil.java |
| package com.example.wordanalysisprogram.util;  public class LogUtil {  public static final int *RESERVED\_TOKEN*=0;  public static final int *VARIABLE*=1;  public static final int *OPERATOR*=2;  public static final int *VALUE*=3;  public static final int *TYPE\_OF\_VAR*=4;   public static String getPureTextLine(String text)  {  return text+"\n";  }   public static String getHeader(int lno)  {  return "============== 第"+lno+"行的分析情况 ==============\n\n";  }   public static String getLog(int lno,String token,int type)  {  StringBuilder sb=new StringBuilder();  sb.append("{ ");  sb.append("第").append(lno).append("行：");  sb.append("当前分析符号为 ").append(token).append(" ， ");  sb.append("类型为 ");   String t="";  switch(type)  {  case *RESERVED\_TOKEN*:  {  t="保留字";  break;  }  case *VARIABLE*:  {  t="变量";  break;  }  case *OPERATOR*:  {  t="算符";  break;  }  case *VALUE*:  {  t="变量值";  break;  }  case *TYPE\_OF\_VAR*:  {  t="类型名称";  break;  }  }  sb.append(t);  sb.append(" }");  return sb.toString();  } } |

4.3.4异常处理

|  |
| --- |
| service/ex/BaseException.java |
| package com.example.wordanalysisprogram.service.ex;  public class BaseException extends Exception{  public BaseException(String cause)  {  super(cause);  }   public BaseException(String cause,int line)  {  super(cause+"，第"+line+"行");  }   public BaseException(String cause,int line, String token)  {  super(cause+"，第"+line+"行，出错的标识符是"+token);  }   public BaseException(String cause,int line,int col)  {  super(cause+"，第"+line+"行，第"+col+"列");  } } |

4.3.5业务层接口和实现

包含两部分，第一步给出符号表，第二步分析代码体

|  |
| --- |
| service/WordAnalysisService.java |
| package com.example.wordanalysisprogram.service;  import com.example.wordanalysisprogram.domain.vo.LogCatVO; import com.example.wordanalysisprogram.domain.Symbol; import com.example.wordanalysisprogram.service.ex.BaseException;  import java.util.List;  public interface WordAnalysisService {   LogCatVO getSymbols(String code) throws BaseException;  LogCatVO checkError(String code, List<Symbol> knownVars, List<String> operators, List<String> reservedTokens) throws BaseException; } |

实现

|  |
| --- |
| service/impl/WordAnalysisServiceImpl.java |
| package com.example.wordanalysisprogram.service.impl;  import android.text.TextUtils; import android.util.Log;   import com.example.wordanalysisprogram.domain.vo.LogCatVO; import com.example.wordanalysisprogram.domain.Symbol; import com.example.wordanalysisprogram.service.WordAnalysisService; import com.example.wordanalysisprogram.service.ex.BaseException; import com.example.wordanalysisprogram.util.IDUtil; import com.example.wordanalysisprogram.util.LogUtil;   import java.util.ArrayList; import java.util.List; import java.util.Locale;  public class WordAnalysisServiceImpl implements WordAnalysisService {   private static final String *TAG* = "WordAnalysisServiceImpl";    boolean isDigit(String token)  {  if(TextUtils.*isEmpty*(token))  {  return false;  }  for(int i=0;i<token.length();i++)  {  char c=token.charAt(i);  if(!Character.*isDigit*(c))  {  return false;  }  }  return true;  }   boolean isVariable(String token)  {  if(TextUtils.*isEmpty*(token))  {  return false;  }  int n=token.length();  for(int i=0;i<n;i++)  {  char c=token.charAt(i);  if(i==1)  {  if(!Character.*isLetter*(c))  {  return false;  }  }  else  {  if(!((Character.*isDigit*(c))||(Character.*isLetter*(c))))  {  return false;  }  }  }  return true;  }   boolean isOperator(String token,List<String> ops)  {  return ops.contains(token);  }   boolean isReservedToken(String token,List<String> ops)  {  return ops.contains(token);  }   @Override  public LogCatVO getSymbols(String code) throws BaseException{   List<String> logs=new ArrayList<>();   List<Symbol> res=new ArrayList<>();   String[] lines=code.split(";");   logs.add(LogUtil.*getPureTextLine*("以下是变量定义分析：\n"));   for(int lno=0;lno<lines.length;lno++)  {  String line=lines[lno];  line=line.trim();  line=line.replaceAll("\n","");  line=line.toLowerCase(Locale.*ROOT*);    if(!TextUtils.*isEmpty*(line))  {  if(line.startsWith("var"))  {  logs.add(LogUtil.*getHeader*(lno));  logs.add(LogUtil.*getLog*(lno,"var",LogUtil.*RESERVED\_TOKEN*));  *//是变量定义语句* if (line.length() <= 3 ||line.charAt(3)!=' '||!line.contains(":")) {  *//todo: 报错* throw new BaseException("var关键字后无变量",lno,"var");   }  *//滤掉var关键字* line=line.substring(3).trim();  String[] tokens=line.split(":");  if(tokens.length!=2)  {  *//todo:报错* throw new BaseException("变量无类型",lno,"var");  }  String[] varNames=tokens[0].split(",");  String type=tokens[1];   if(!Symbol.*types*.contains(type))  {  *//todo:报错* throw new BaseException("变量类型不正确",lno,type);  }  for(int i=0;i<varNames.length;i++)  {  Log.*d*("TAG", "getSymbols: 变量名是 "+varNames[i]);   if (!(isVariable(varNames[i])))  {  *//todo:报错* throw new BaseException("变量名非法",lno,varNames[i]);  }   Symbol vo=new Symbol();  vo.setId(IDUtil.*getUUID*());  vo.setName(varNames[i]);  vo.setType(type);  if(res.contains(vo))  {  throw new BaseException("变量重复定义",lno,varNames[i]);  }  logs.add(LogUtil.*getLog*(lno,varNames[i],LogUtil.*VARIABLE*));   res.add(vo);  }  logs.add(LogUtil.*getLog*(lno,type,LogUtil.*TYPE\_OF\_VAR*));  }  else  {  continue;*//防止变量分开定义* }  }   }   LogCatVO logCatVO=new LogCatVO();  logCatVO.setType(LogCatVO.*VARIABLE\_ANALYSIS*);   logCatVO.setSymbols(res);  logCatVO.setLogs(logs);   return logCatVO;  }   @Override  public LogCatVO checkError(String code, List<Symbol> knownVars, List<String> operators, List<String> reservedTokens) throws BaseException  {  LogCatVO logCatVO=new LogCatVO();  logCatVO.setType(LogCatVO.*CODE\_ANALYSIS*);    List<String> logs=new ArrayList<>();   logs.add(LogUtil.*getPureTextLine*("以下是代码段分析：\n"));   String[] lines=code.split(";");  for(int lno=0;lno<lines.length;lno++)  {  String line=lines[lno];  line=line.trim();  line=line.replaceAll("\n","");  line=line.toLowerCase(Locale.*ROOT*);  *//System.out.println("代码行："+line);* if(!TextUtils.*isEmpty*(line))  {  if(line.startsWith("var"))  {  continue;  }   logs.add(LogUtil.*getHeader*(lno));  String[] tokens=line.split("\\s+");  for(String token:tokens)  {  token=token.trim();  System.*out*.println("当前分析的标识符是："+token);    if(isOperator(token,operators))  {  logs.add(LogUtil.*getLog*(lno,token,LogUtil.*OPERATOR*));  }  else if(isReservedToken(token,reservedTokens))  {  logs.add(LogUtil.*getLog*(lno,token,LogUtil.*RESERVED\_TOKEN*));  }  else  {  *//不是保留字，是变量* if(isDigit(token))  {  *//System.out.println("是数字！");* logs.add(LogUtil.*getLog*(lno,token,LogUtil.*VALUE*));   }  else if(isVariable(token))  {  Symbol vo=new Symbol();  vo.setName(token);  if(!knownVars.contains(vo))  {  throw new BaseException("变量未声明就使用了",lno,token);  }   logs.add(LogUtil.*getLog*(lno,token,LogUtil.*VARIABLE*));  }  else  {  throw new BaseException("非法的标识符",lno,token);  }  }  }  }   }   logCatVO.setFlag(true);  logCatVO.setLogs(logs);   return logCatVO;  } } |

4.4界面层

4.4.1主界面

界面描述

|  |
| --- |
| res/layout/activity\_main.xml |
| *<?*xml version="1.0" encoding="utf-8"*?>* <LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"  xmlns:app="http://schemas.android.com/apk/res-auto"  android:orientation="vertical"  xmlns:tools="http://schemas.android.com/tools"  android:layout\_width="match\_parent"  android:layout\_height="match\_parent"  tools:context=".MainActivity">   <TextView  android:layout\_margin="5px"  android:textAlignment="center"  android:layout\_width="match\_parent"  android:layout\_height="wrap\_content"  android:text="词法分析演示程序"  />  <TextView  android:layout\_width="match\_parent"  android:layout\_height="wrap\_content"  android:layout\_margin="5px"  android:text="输入一段代码，我们会自动生成符号表，并告诉您错误（如果有）"  />  <EditText  android:layout\_margin="5px"  android:id="@+id/text\_code\_segment"  android:layout\_width="match\_parent"  android:layout\_height="0px"  android:layout\_weight="1"  android:maxLines="255"  android:allowUndo="true"  android:hint="请输入代码"  android:textColor="#000000"  android:gravity="top"  android:background="#a9a9a9"  />   <Button  android:layout\_margin="5px"  android:id="@+id/btn\_check"  android:layout\_width="match\_parent"  android:layout\_height="wrap\_content"  android:text="立即检查！"  /> </LinearLayout> |

主界面控制代码

|  |
| --- |
| MainActivity.java |
| package com.example.wordanalysisprogram;  import androidx.appcompat.app.AppCompatActivity;  import android.content.Intent; import android.os.Bundle; import android.widget.Button; import android.widget.EditText;  import com.example.wordanalysisprogram.util.IDUtil;  public class MainActivity extends AppCompatActivity {   private static final String *TAG* = "MainActivity";   private EditText textCodeSegment;  private Button btnCheck;   @Override  protected void onCreate(Bundle savedInstanceState) {  super.onCreate(savedInstanceState);  setContentView(R.layout.*activity\_main*);  System.*out*.println(IDUtil.*getUUID*());   textCodeSegment=findViewById(R.id.*text\_code\_segment*);  btnCheck=findViewById(R.id.*btn\_check*);   btnCheck.setOnClickListener((e)->{  String code=textCodeSegment.getText().toString();  Intent intent=new Intent(MainActivity.this,ResultActivity.class);  intent.putExtra("code",code);  startActivity(intent);  });  } } |

4.4.2符号表显示控件

界面描述

|  |
| --- |
| res/layout/symbol\_item.xml |
| *<?*xml version="1.0" encoding="utf-8"*?>* <LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"  android:layout\_width="match\_parent"  android:layout\_height="wrap\_content">   <TextView  android:id="@+id/text\_id"  android:layout\_width="0px"  android:layout\_weight="1"  android:layout\_height="wrap\_content"  android:layout\_marginRight="5px"/>   <TextView  android:id="@+id/text\_var\_name"  android:layout\_width="0px"  android:layout\_weight="1"  android:layout\_height="wrap\_content"  android:layout\_marginRight="5px"/>   <TextView  android:id="@+id/text\_var\_type"  android:layout\_width="0px"  android:layout\_weight="1"  android:layout\_height="wrap\_content"  android:layout\_marginRight="5px"/> </LinearLayout> |

控件实现代码

|  |
| --- |
| ui/adapter/SymbolAdapter.java |
| package com.example.wordanalysisprogram.ui.adapter;  import android.content.Context; import android.view.LayoutInflater; import android.view.View; import android.view.ViewGroup; import android.widget.ArrayAdapter; import android.widget.TextView;  import androidx.annotation.NonNull; import androidx.annotation.Nullable;  import com.example.wordanalysisprogram.R; import com.example.wordanalysisprogram.domain.Symbol;  import java.util.List;  public class SymbolAdapter extends ArrayAdapter<Symbol> {   private int resId;   public SymbolAdapter(@NonNull Context context, int resource, @NonNull List<Symbol> objects) {  super(context, resource, objects);  resId=resource;  }   @NonNull  @Override  public View getView(int position, @Nullable View convertView, @NonNull ViewGroup parent) {  Symbol vo=getItem(position);   View view= LayoutInflater.*from*(getContext()).inflate(resId,parent,false);  TextView txtId=view.findViewById(R.id.*text\_id*);  TextView txtName=view.findViewById(R.id.*text\_var\_name*);  TextView textType=view.findViewById(R.id.*text\_var\_type*);   txtId.setText(vo.getId());  txtName.setText(vo.getName());  textType.setText(vo.getType());   return view;  } } |

4.2.3结果展示界面

界面描述

|  |
| --- |
| res/layout/activity\_result.xml |
| *<?*xml version="1.0" encoding="utf-8"*?>* <LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"  xmlns:app="http://schemas.android.com/apk/res-auto"  android:orientation="vertical"  xmlns:tools="http://schemas.android.com/tools"  android:layout\_width="match\_parent"  android:layout\_height="match\_parent"  tools:context=".ResultActivity">    <TextView  android:layout\_margin="5px"  android:layout\_width="match\_parent"  android:layout\_height="wrap\_content"  android:textSize="22dp"  android:text="符号表："/>     <LinearLayout  android:layout\_margin="5px"  android:layout\_width="match\_parent"  android:layout\_height="wrap\_content">  <TextView  android:layout\_width="0px"  android:layout\_weight="1"  android:layout\_height="wrap\_content"  android:textAlignment="center"  android:text="变量ID"/>  <TextView  android:layout\_width="0px"  android:layout\_weight="1"  android:layout\_height="wrap\_content"  android:textAlignment="center"  android:text="变量名"/>  <TextView  android:layout\_width="0px"  android:layout\_weight="1"  android:layout\_height="wrap\_content"  android:textAlignment="center"  android:text="变量类型"/>  </LinearLayout>    <View  android:layout\_width="match\_parent"  android:layout\_height="5dp"  android:background="#00FF00"  />   <ListView  android:layout\_margin="5px"  android:id="@+id/list\_vars"  android:layout\_width="match\_parent"  android:background="#a9a9a9"  android:layout\_height="0px"  android:layout\_weight="1.5"/>   <LinearLayout  android:orientation="horizontal"  android:layout\_width="match\_parent"  android:layout\_height="0px"  android:layout\_weight="1">  <LinearLayout  android:layout\_width="0px"  android:layout\_height="match\_parent"  android:orientation="vertical"  android:layout\_weight="1">   <TextView  android:layout\_margin="5px"  android:layout\_width="match\_parent"  android:layout\_height="wrap\_content"   android:textSize="22dp"  android:text="词法分析日志："/>   <View  android:layout\_width="match\_parent"  android:layout\_height="5dp"  android:background="#00FF00"  />   <EditText  android:layout\_margin="5px"  android:textColor="#000000"  android:id="@+id/text\_log\_list"  android:layout\_width="match\_parent"  android:layout\_height="0px"  android:layout\_weight="0.5"  android:singleLine="false"  android:maxLines="255"  android:background="#a9a9a9"  android:gravity="top"  android:inputType= "textMultiLine"/>    </LinearLayout>   <View  android:layout\_width="5dp"  android:layout\_height="match\_parent"  android:background="#00FF00"  />   <LinearLayout  android:layout\_width="0px"  android:layout\_height="match\_parent"  android:layout\_weight="1"  android:orientation="vertical">  <TextView  android:layout\_margin="5px"  android:layout\_width="match\_parent"  android:layout\_height="wrap\_content"   android:textSize="22dp"  android:text="错误报告："/>   <View  android:layout\_width="match\_parent"  android:layout\_height="5dp"  android:background="#00FF00"  />   <EditText  android:layout\_margin="5px"  android:textColor="#000000"  android:id="@+id/text\_exception\_list"  android:layout\_width="match\_parent"  android:layout\_height="0px"  android:layout\_weight="0.5"  android:maxLines="255"  android:background="#a9a9a9"  android:gravity="top"  android:inputType= "textMultiLine"/>  </LinearLayout>  </LinearLayout>    <Button  android:layout\_margin="5px"  android:id="@+id/btn\_back"  android:layout\_width="match\_parent"  android:layout\_height="wrap\_content"  android:text="返回编辑界面"/> </LinearLayout> |

实现代码

|  |
| --- |
| ResultActivity.java |
| package com.example.wordanalysisprogram;  import androidx.appcompat.app.AppCompatActivity;  import android.content.Intent; import android.os.Bundle; import android.util.Log; import android.widget.Button; import android.widget.EditText; import android.widget.ListView;  import com.example.wordanalysisprogram.domain.vo.LogCatVO; import com.example.wordanalysisprogram.domain.Symbol; import com.example.wordanalysisprogram.service.WordAnalysisService; import com.example.wordanalysisprogram.service.ex.BaseException; import com.example.wordanalysisprogram.service.impl.WordAnalysisServiceImpl; import com.example.wordanalysisprogram.ui.adapter.SymbolAdapter; import com.example.wordanalysisprogram.util.PropertiesUtil;  import java.util.List;  public class ResultActivity extends AppCompatActivity {   private static final String *TAG* = "ResultActivity";   private String code;   private ListView varView;   private EditText txtException;   private EditText txtLog;   private List<Symbol> res;   private Button btnBack;    private WordAnalysisService service=new WordAnalysisServiceImpl();   @Override  protected void onCreate(Bundle savedInstanceState) {  super.onCreate(savedInstanceState);  setContentView(R.layout.*activity\_result*);   varView=findViewById(R.id.*list\_vars*);  txtException=findViewById(R.id.*text\_exception\_list*);  txtLog=findViewById(R.id.*text\_log\_list*);  btnBack=findViewById(R.id.*btn\_back*);   btnBack.setOnClickListener((e)->{  finish();  });    Intent intent=getIntent();   code=intent.getStringExtra("code");   Log.*d*(*TAG*,code);   List<String> operators= PropertiesUtil.*getOperators*(this);  List<String> reservedTokens=PropertiesUtil.*getReservedTokens*(this);    try {  LogCatVO ret = service.getSymbols(code);  if(ret.getType()==LogCatVO.*VARIABLE\_ANALYSIS*)  {  res=ret.getSymbols();  List<String> logs=ret.getLogs();   for(Symbol vo:res)  {  Log.*d*(*TAG*,vo.toString());  }   SymbolAdapter adapter=new SymbolAdapter(ResultActivity.this,R.layout.*symbol\_item*,res);  varView.setAdapter(adapter);   StringBuilder sb=new StringBuilder();  for (String log:logs)  {  sb.append(log+"\n");  }   txtLog.setText(txtLog.getText().toString()+"\n"+sb.toString()+"\n");   }  else  {  throw new BaseException("未知错误");  }   LogCatVO ret2=service.checkError(code,res,operators,reservedTokens);  if(ret2.getType()==LogCatVO.*CODE\_ANALYSIS*)  {  boolean flag=ret2.isFlag();  if(flag)  {  txtException.setText("恭喜，词法分析通过！");  }   List<String> logs=ret2.getLogs();  StringBuilder sb=new StringBuilder();  for (String log:logs)  {  sb.append(log+"\n");  }   txtLog.setText(txtLog.getText().toString()+"\n"+sb.toString());  }  else  {  throw new BaseException("未知错误");  }   } catch (BaseException e) {  e.printStackTrace();  String cause=e.getMessage();  txtException.setText(cause);  }  } } |

4.5工程配置文件

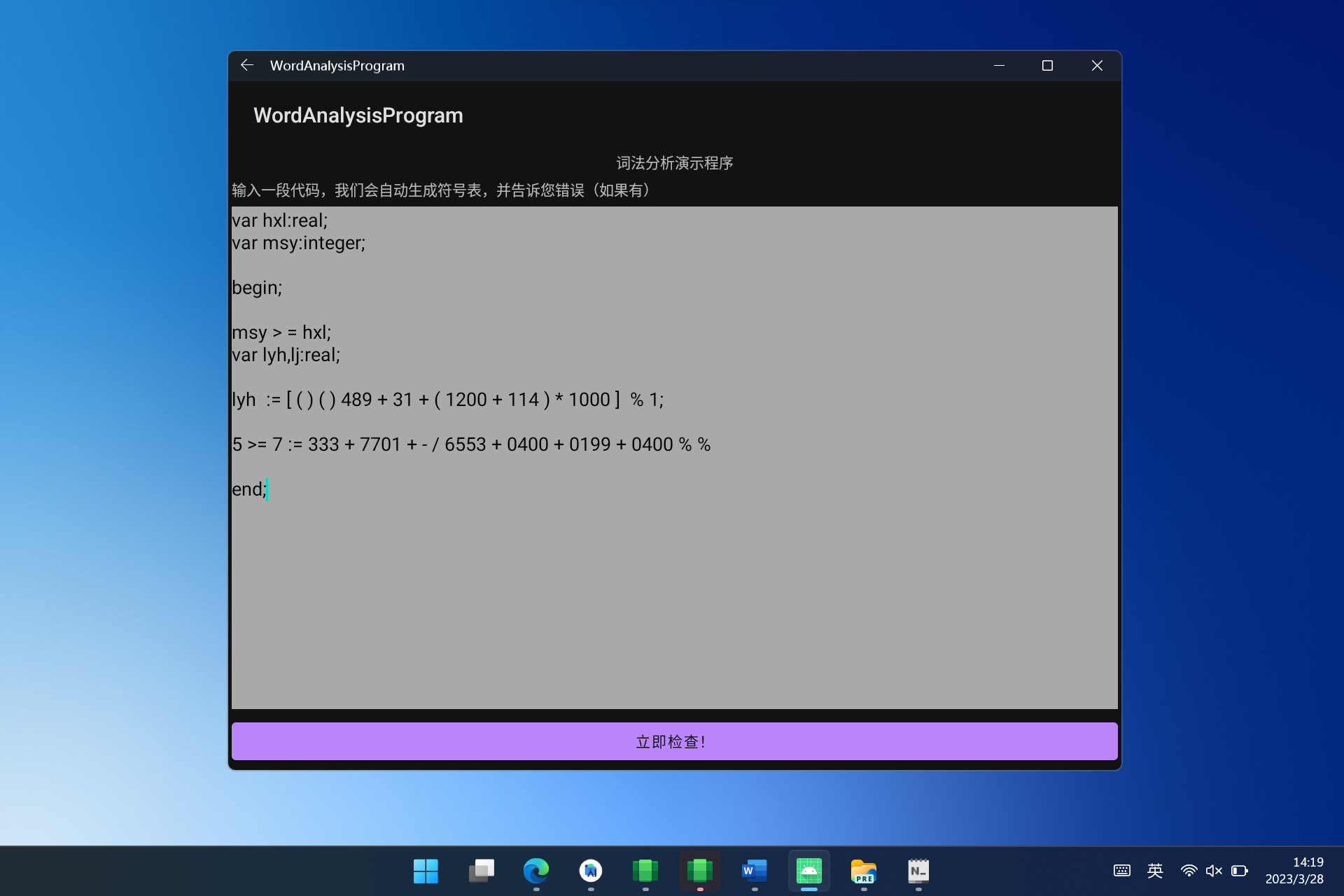
|  |
| --- |
| AndroidMainfist.xml |
| *<?*xml version="1.0" encoding="utf-8"*?>* <manifest xmlns:android="http://schemas.android.com/apk/res/android"  xmlns:tools="http://schemas.android.com/tools">   <application  android:allowBackup="true"  android:dataExtractionRules="@xml/data\_extraction\_rules"  android:fullBackupContent="@xml/backup\_rules"  android:icon="@mipmap/ic\_launcher"  android:label="@string/app\_name"  android:roundIcon="@mipmap/ic\_launcher\_round"  android:supportsRtl="true"  android:theme="@style/Theme.WordAnalysisProgram"  tools:targetApi="31">  <activity  android:name=".ResultActivity"  android:exported="false">  <meta-data  android:name="android.app.lib\_name"  android:value="" />  </activity>  <activity  android:name=".MainActivity"  android:exported="true">  <intent-filter>  <action android:name="android.intent.action.MAIN" />   <category android:name="android.intent.category.LAUNCHER" />  </intent-filter>   <meta-data  android:name="android.app.lib\_name"  android:value="" />  </activity>  </application>  </manifest> |

**5.运行效果**

输入下面的代码

|  |
| --- |
| Input.txt |
| var hxl:real;  var msy:integer;  begin;  msy > = hxl;  var lyh,lj:real;  lyh := [ ( ) ( ) 489 + 31 + ( 1200 + 114 ) \* 1000 ] % 1;  5 >= 7 := 333 + 7701 + - / 6553 + 0400 + 0199 + 0400 % %  end; |

运行结果



检查结果

图形用户界面, 应用程序

描述已自动生成

接下来展示几个错误的例子

变量未定义

图形用户界面, 文本

描述已自动生成

图形用户界面, 应用程序

描述已自动生成

非法变量名

电脑萤幕的截图

描述已自动生成

电脑萤幕的截图

描述已自动生成

变量类型不正确

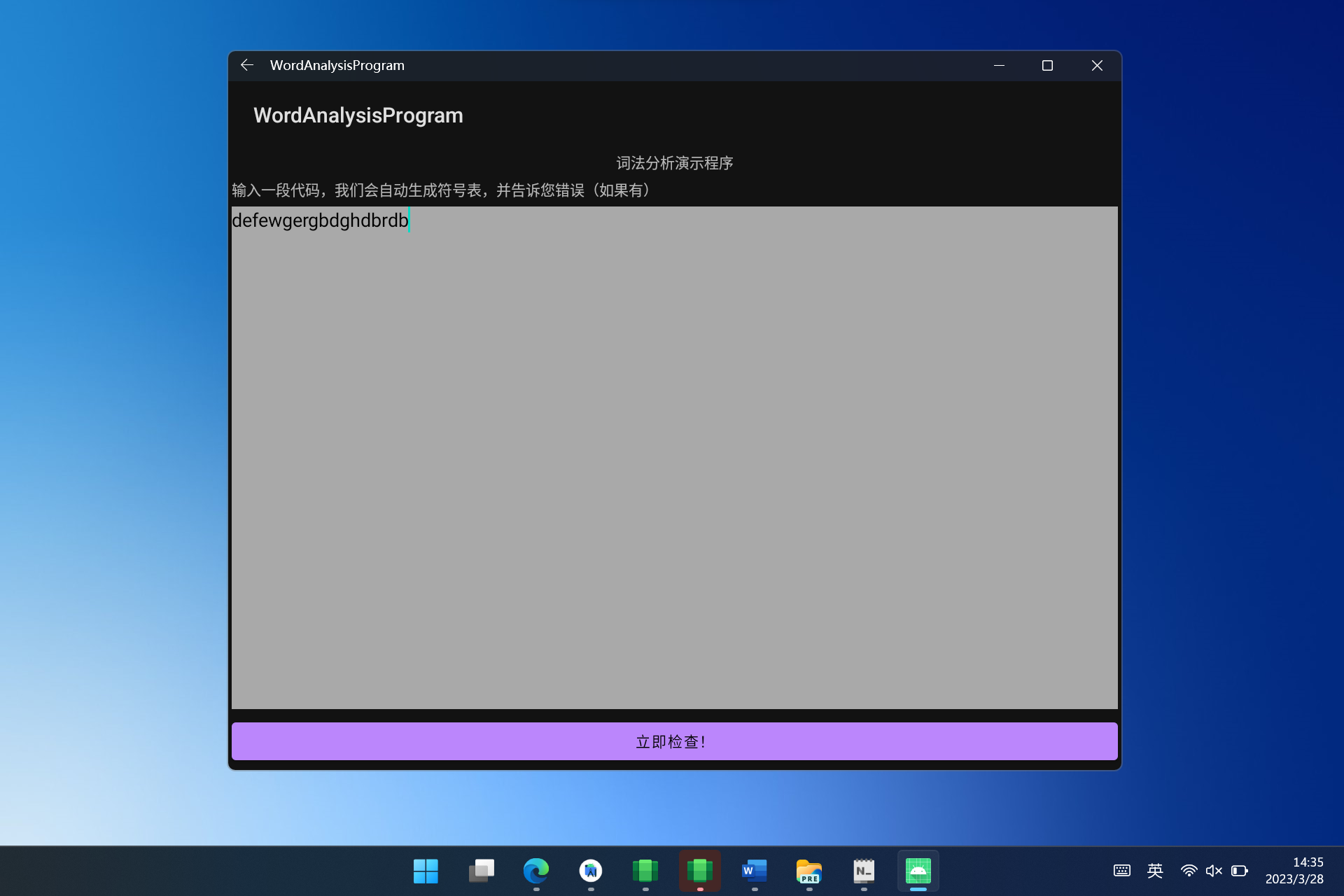
图形用户界面, 文本

描述已自动生成

电脑萤幕的截图

描述已自动生成

胡言乱语



电脑萤幕的截图

描述已自动生成

他不认识乱码，所以报变量名未定义