实验二 语法分析

**实验目的**

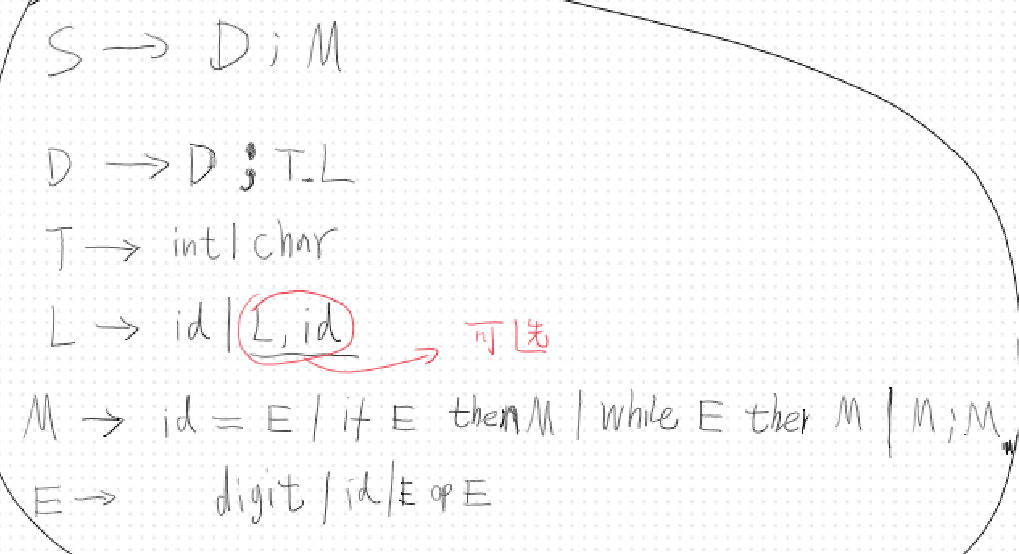
掌握求解first集合和follow集合的方法

掌握自底向上规约的语法分析方式，会根据产生式设计action和goto表

会判别文法类型，会用规约法分析LR（0~1），SLR（1）文法

**实验要求**

已知文法如下，求action和goto表，设计语法分析工具



**实验方法和步骤**

根据文法写出自动机并求解action和goto表

由于这个文法实在是太大，人工求解费时费力。因此我们选取了一个子集进行演示。但是其分析思路和实验原理，无论使用什么文法都是万变不离其宗的。

这个文法的子集如下：

E->E op T

E->T

T->T op F

T->F

F->(E)

F->id

为了便于演示，这里的运算符只有+和\*两种，标识符统一用i代表，事实上标识符的确定是通过词法分析完成的。

此文法的带item标记的DFA如图所示

根据算法，构造的action和goto表如图所示：（按照上面的顺序对产生式进行编号）

| 已知状态  和预测 | Action | | | | | | Goto | | |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| + | \* | （ | ） | Id | $（输入结束） | E | T | F |
| 0 |  |  | S4 |  | S5 |  | S1 | S2 | S3 |
| 1 | S6 |  |  |  |  | ACC |  |  |  |
| 2 | R2 | S7 |  | R2 |  | R2 |  |  |  |
| 3 | R4 | R4 |  | R4 |  | R4 |  |  |  |
| 4 |  |  | S4 |  | S5 |  | S8 | S2 | S3 |
| 5 | R6 | R6 |  | R6 |  | R6 |  |  |  |
| 6 |  |  | S4 |  | S5 |  |  | S9 | S3 |
| 7 |  |  | S4 |  | S5 |  |  |  | S10 |
| 8 | S6 |  |  | S11 |  |  |  |  |  |
| 9 | R1 | S7 |  | R1 |  | R1 |  |  |  |
| 10 | R3 | R3 |  | R3 | R3 |  |  |  |  |
| 11 | R5 | R5 |  | R5 | R5 |  |  |  |  |

显然这是SLR（1）文法

项目结构如图所示：

图形用户界面, 文本, 应用程序, 聊天或短信

描述已自动生成

编译环境：JDK1.8，Android SDK 13（API33）

运行环境：JRE1.8，Android Runtime 13（API33）

程序代码：

1数据对象

|  |
| --- |
| vo\ResultVO.java |
| package com.example.syntaxanalyser.vo;  public class ResultVO {  public static final int *PASS*=1;  public static final int *FAIL*=0;   private int result;  private String log;   public ResultVO(){}   public ResultVO(int result, String log) {  this.result = result;  this.log = log;  }   public int getResult() {  return result;  }   public void setResult(int result) {  this.result = result;  }   public String getLog() {  return log;  }   public void setLog(String log) {  this.log = log;  } } |

2数据处理

接口

|  |
| --- |
| service\ProductionService.java |
| package com.example.syntaxanalyser.service;  public interface ProductionService {  String getProduction(int i);  int getNumOfP(int i); } |

|  |
| --- |
| service\SyntaxAnalysisService.java |
| package com.example.syntaxanalyser.service;  import com.example.syntaxanalyser.vo.ResultVO;  public interface SyntaxAnalysisService {  ResultVO getResult(String input); } |

实现方式

|  |
| --- |
| service\impl\ProductionServiceImpl.java |
| package com.example.syntaxanalyser.service.impl;  import com.example.syntaxanalyser.service.ProductionService;  import java.util.Vector;  */\*\*  \** ***@author*** *SteveWorkshop  \*  \*/* public class ProductionServiceImpl implements ProductionService {    private String[] productions={"S'->E","E->E+T","E->T","T->T\*F","T->F","F->(E)","F->i"};  private int[] numOfP={1,3,1,3,1,3,1};  private Vector<String> productionList=new Vector<>();   public ProductionServiceImpl() {  for(String st:productions)  productionList.addElement(st);  }   @Override  public String getProduction(int i){  return productions[i];  }   @Override  public int getNumOfP(int i){  return numOfP[i];  } } |

|  |
| --- |
| service\impl\SyntaxServiceImpl.java |
| package com.example.syntaxanalyser.service.impl;  import com.example.syntaxanalyser.service.ProductionService; import com.example.syntaxanalyser.service.SyntaxAnalysisService; import com.example.syntaxanalyser.vo.ResultVO;  import java.io.BufferedReader; import java.io.BufferedWriter; import java.io.FileReader; import java.io.FileWriter; import java.io.IOException; import java.util.LinkedList; import java.util.Stack; import java.util.Vector;  */\*\*  \*   \*/  /\*\*  \** ***@author*** *zhongfang  \*   \*/* public class SyntaxAnalysisServiceImpl implements SyntaxAnalysisService {   private String temp1;  private LinkedList<Character> list1 = new LinkedList<>();  private Stack<Character> stack = new Stack<>();  private Stack<Integer> stack2 = new Stack<>();  private ProductionService productionService = new ProductionServiceImpl();  private String actions = "";  private Vector<Integer> sequenceOfP=new Vector<>();   private String[][] table = { { "", "", "S4", "", "S5", "", "1", "2", "3" }, // 0  { "S6", "", "", "", "", "accept", "", "", "" }, // 1  { "r2", "S7", "", "r2", "", "r2", "", "", "" }, // 2  { "r4", "r4", "", "r4", "", "r4", "", "", "" },// 3  { "", "", "S4", "", "S5", "", "8", "2", "3" },// 4  { "r6", "r6", "", "r6", "", "r6", "", "", "" },// 5  { "", "", "S4", "", "S5", "", "", "9", "3" },// 6  { "", "", "S4", "", "S5", "", "", "", "10" },// 7  { "S6", "", "", "S11", "", "", "", "", "" },// 8  { "r1", "S7", "", "r1", "", "r1", "", "", "" },// 9  { "r3", "r3", "", "r3", "", "r3", "", "", "" },// 10  { "r5", "r5", "", "r5", "", "r5", "", "", "" },// 11  };   @Override  public ResultVO getResult(String input) {  char a[];   ResultVO ret=new ResultVO();   StringBuilder logs=new StringBuilder();   stack.clear();  list1.clear();  stack2.clear();  sequenceOfP.clear();  temp1 = input.trim();  temp1.replaceAll("\\s+", "");  a = temp1.toCharArray();  for (char \_char : a) {  list1.offer(\_char);  }  list1.offerLast('$');  stack.push('$');  stack2.push(0);    logs.append("状态\t 符号\t 输入\t 动作\n");   ResultVO b = analysis();   String l = b.getLog();  logs.append(l);  if (b.getResult()==ResultVO.*PASS*){   logs.append("这个输入符合设定的SLR(1)文法\n");  logs.append("产生式顺序如下：(bottom-up)");    ret.setResult(ResultVO.*PASS*);  for(int i=0;i<sequenceOfP.size();i++){  Integer n=sequenceOfP.get(i);   logs.append(productionService.getProduction(n)+"\n");   }   }  else  {  ret.setResult(ResultVO.*FAIL*);  logs.append("这个输入不符合设定的SLR(1)文法\n");  }  logs.append("-----------------------------------------");     ret.setLog(logs.toString());  return ret;  }   private int getOrder(char c) {  if (c == '+')  return 0;  else if (c == '\*')  return 1;  else if (c == '(')  return 2;  else if (c == ')')  return 3;  else if (c == 'i')  return 4;  else if (c == '$')  return 5;  else if (c == 'E')  return 6;  else if (c == 'T')  return 7;  else if(c=='F')  return 8;  else   return -1;  }   private String display() {  String symbols = "";  String states = "";  String input = "";   StringBuilder sb=new StringBuilder();   Object[] symbolObjects = stack.toArray();  for (int i = 0; i < symbolObjects.length; i++) {  symbols += symbolObjects[i].toString();   }   Object[] stateObjects = stack2.toArray();  for (int i = 0; i < stack2.size(); i++) {  states += stateObjects[i].toString();  }   Object[] inputObjects = list1.toArray();  for (int i = 0; i < list1.size(); i++) {  input += inputObjects[i].toString();  }   //写日志  sb.append(states + "\t" + symbols + "\t" + input + "\t"  + actions);   return sb.toString();  }    private ResultVO analysis() {   ResultVO vo=new ResultVO();  StringBuilder sb=new StringBuilder();   while (true) {  actions = "";  char c = list1.peekFirst();  int i = getOrder(c);  if(i==-1)  {  vo.setResult(ResultVO.*FAIL*);  vo.setLog(sb.toString());  return vo;  }  String string2 = table[stack2.peek()][i];  if (string2.trim().equals("".trim()))  {  vo.setResult(ResultVO.*FAIL*);  vo.setLog(sb.toString());  return vo;  }  else if (string2.equals("accept")) {  actions+="接收";  String log = display();  //System.out.println("黄欣灵："+log);  sb.append(log+"\n");   vo.setResult(ResultVO.*PASS*);  vo.setLog(sb.toString());  return vo;//OK  }  else if (string2.charAt(0) == 'S')  {  String s = string2.substring(1);  int n = Integer.*parseInt*(s);  //System.out.println("Shift " + s);   actions += "移入，状态跳转到： " + s;  String log = display();   //System.out.println("黄欣灵："+log);   sb.append(log+"\n");  list1.pollFirst();  stack2.push(n);  stack.push(c);   }  else if (string2.charAt(0) == 'r')  {  String s = string2.substring(1);  int n = Integer.*parseInt*(s);  //System.out.println("Reduce " + s);  actions += "使用产生式 " + s+" 规约";  String log = display();  //System.out.println("黄欣灵："+log);  sb.append(log+"\n");  sequenceOfP.addElement(n);   int n2 = productionService.getNumOfP(n);  for (int i1 = 0; i1 < n2; i1++) {  stack.pop();  stack2.pop();  }   char \_char1 = productionService.getProduction(n).charAt(0);  stack.push(\_char1);  String s1 = table[stack2.peek()][getOrder(\_char1)];  if (s1.trim().equals(""))  {  vo.setResult(ResultVO.*FAIL*);  vo.setLog(sb.toString());  return vo;  }  else  {  stack2.push(Integer.*parseInt*(s1));  }  }  }  } } |

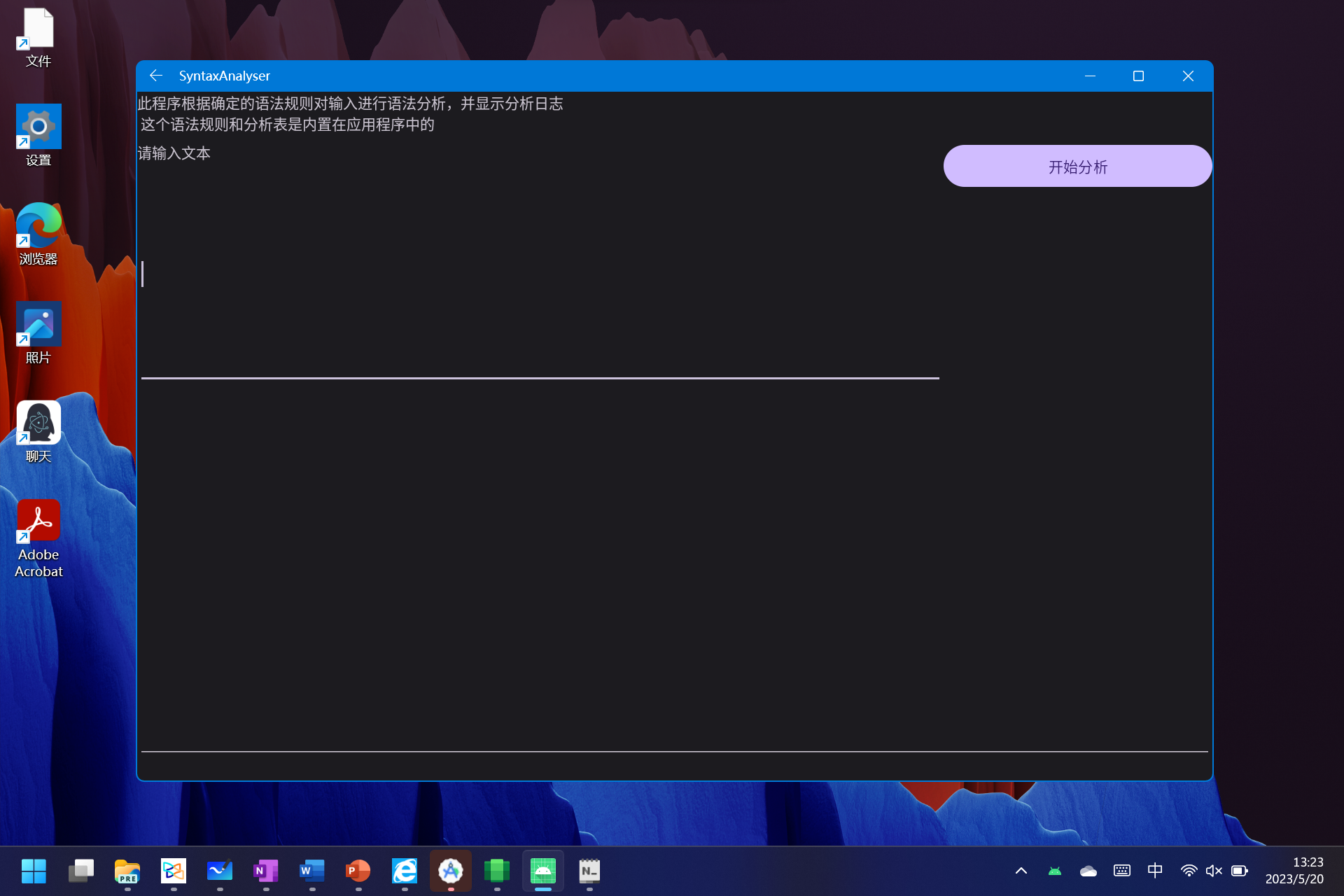
UI设计：

|  |
| --- |
| 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"  xmlns:tools="http://schemas.android.com/tools"  android:orientation="vertical"  android:layout\_width="match\_parent"  android:layout\_height="match\_parent"  tools:context=".MainActivity">   <LinearLayout  android:orientation="vertical"  android:layout\_width="match\_parent"  android:layout\_height="0px"  android:layout\_weight="1">   <TextView  android:layout\_width="match\_parent"  android:layout\_height="wrap\_content"  android:text="此程序根据确定的语法规则对输入进行语法分析，并显示分析日志\n 这个语法规则和分析表是内置在应用程序中的"  />  </LinearLayout>   <LinearLayout  android:orientation="horizontal"  android:layout\_width="match\_parent"  android:layout\_height="0px"  android:layout\_weight="5">   <LinearLayout  android:orientation="vertical"  android:layout\_width="0px"  android:layout\_height="match\_parent"  android:layout\_weight="3">  <TextView  android:layout\_width="match\_parent"  android:text="请输入文本"  android:layout\_height="wrap\_content"/>  <EditText  android:id="@+id/txt\_input"  android:inputType="textMultiLine"  android:layout\_width="match\_parent"  android:layout\_height="match\_parent"/>  </LinearLayout>   <LinearLayout  android:layout\_width="0px"  android:layout\_height="match\_parent"  android:layout\_weight="1"  android:orientation="vertical">  <Button  android:text="开始分析"  android:id="@+id/btn\_generate"  android:layout\_width="match\_parent"  android:layout\_height="wrap\_content"/>  </LinearLayout>  </LinearLayout>  <LinearLayout  android:orientation="vertical"  android:layout\_width="match\_parent"  android:layout\_height="0px"  android:layout\_weight="8">  <EditText  android:id="@+id/txt\_result"  android:layout\_width="match\_parent"  android:layout\_height="0px"  android:inputType="textMultiLine"  android:layout\_weight="1"/>  <TextView  android:id="@+id/txt\_status"  android:layout\_width="match\_parent"  android:layout\_height="wrap\_content"/>  </LinearLayout>  </LinearLayout> |

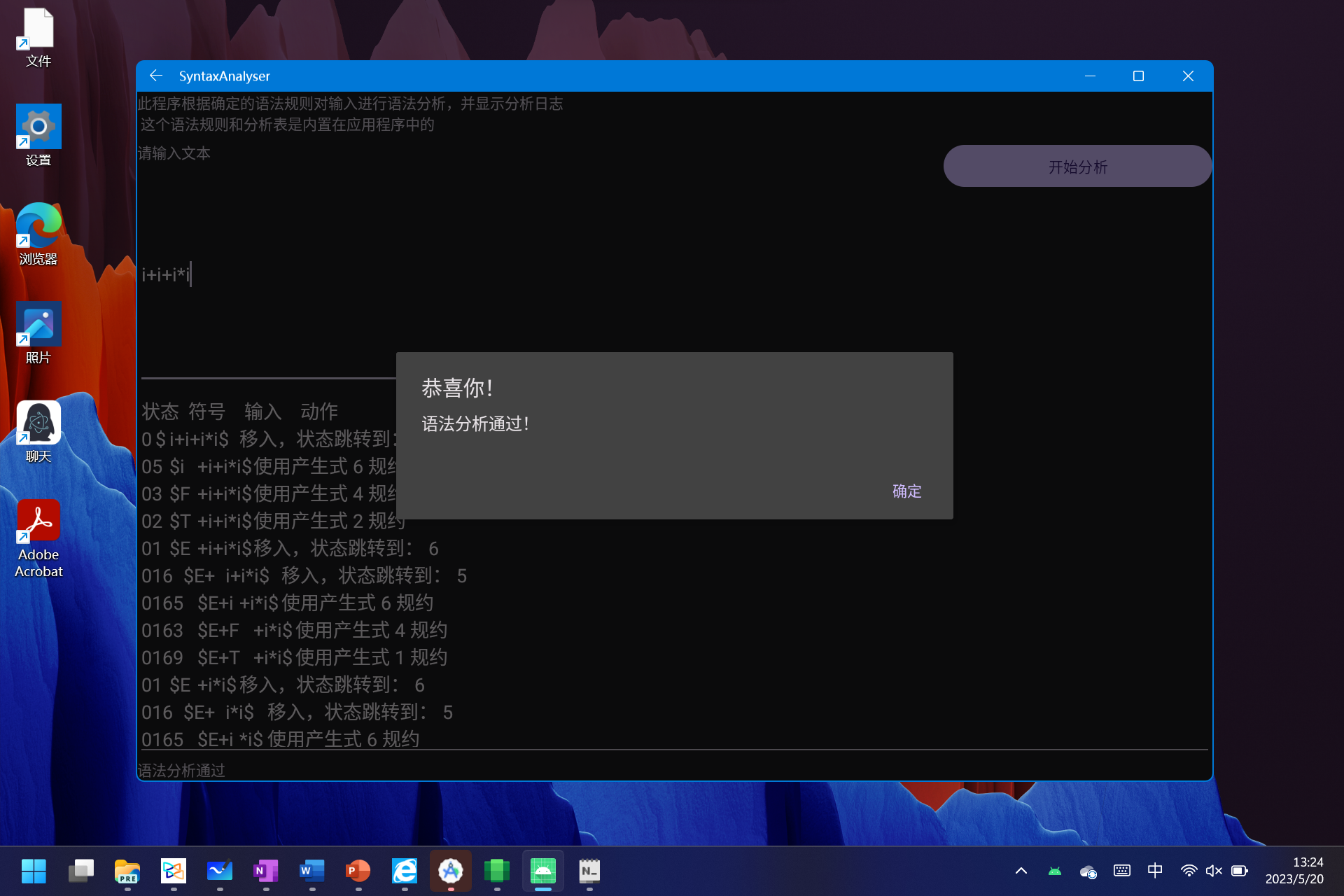
主界面activity

|  |
| --- |
| MainActivity.java |
| package com.example.syntaxanalyser;  import androidx.appcompat.app.AlertDialog; import androidx.appcompat.app.AppCompatActivity;  import android.content.DialogInterface; import android.os.Bundle; import android.widget.Button; import android.widget.EditText; import android.widget.TextView;  import com.example.syntaxanalyser.service.SyntaxAnalysisService; import com.example.syntaxanalyser.service.impl.SyntaxAnalysisServiceImpl; import com.example.syntaxanalyser.vo.ResultVO;  public class MainActivity extends AppCompatActivity {   private EditText txtInput;  private Button btnGenerate;  private EditText txtResult;  private TextView txtStatus;   private SyntaxAnalysisService service=new SyntaxAnalysisServiceImpl();   @Override  protected void onCreate(Bundle savedInstanceState) {  super.onCreate(savedInstanceState);  setContentView(R.layout.*activity\_main*);   txtInput=findViewById(R.id.*txt\_input*);  btnGenerate=findViewById(R.id.*btn\_generate*);  txtResult=findViewById(R.id.*txt\_result*);  txtStatus=findViewById(R.id.*txt\_status*);   btnGenerate.setOnClickListener((e)->{  String input=txtInput.getText().toString();  ResultVO vo=service.getResult(input);  AlertDialog.Builder builder=new AlertDialog.Builder(MainActivity.this);  String logs=vo.getLog();  txtResult.setText(logs);   if(vo.getResult()==ResultVO.*PASS*)  {  txtStatus.setText("语法分析通过");  builder.setTitle("恭喜你！");  builder.setMessage("语法分析通过！");  }  else  {  txtStatus.setText("语法分析未通过");  builder.setTitle("很遗憾！");  builder.setMessage("语法分析未通过！请检查日志");  }  builder.setCancelable(false);  builder.setPositiveButton("确定", new DialogInterface.OnClickListener() {  @Override  public void onClick(DialogInterface dialog, int which) {   }  });  builder.show();  });  } } |

**运行结果：**



输入：i+i+i\*i



输入一个错误的输入，结果如下：

电脑萤幕的截图

描述已自动生成