**编译原理 实验报告**

|  |  |  |
| --- | --- | --- |
| **评 语（4号楷体）** | **成绩** |  |
| 教 师： 邓岳  年 月 日 | | |

**教学班级： 1903052**

**学生学号： 19030500122**

**学生姓名： 赵书晨**

**实验日期： 2021.12**

**一、实验目的**

通过做上机题加深对编译器构造原理和方法的理解，巩固所学知识。

1. 会用正规式设计简单语言的词法；

2. 会用产生式设计简单语言的语法；

3. 会用递归下降子程序编写语言的解释器。

**二、实验环境**

OS: Windows10

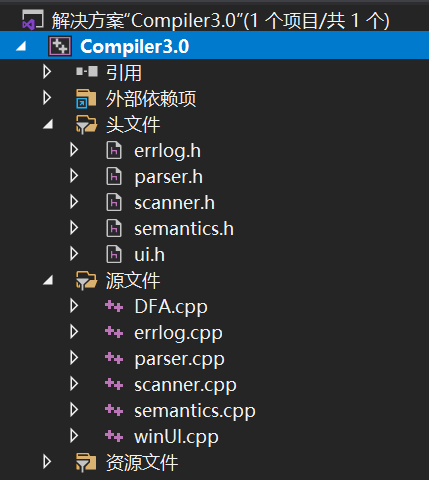
IDE: Visual Studio 2019

### **三、实验内容**

**总体结构：**

编译采用一趟扫描,以语法分析为核心,词法分析,语义分析和代码生成程序都作为一个独立的过程,当语法分析需要读入单词的时候就调用词法分析程序,而语法分析与语义分析正确需生成相应的目标代码时,则调用代码生成程序。用出错处理程序对语法,此法和语义分析遇到的错误在给出源程序中出错的位置和错误性质。当程序编译正确时,编译程序自动调用解释执行程序,对目标代码进行解释执行,并按照输入输出运行结果。

**模块划分：**

　　Scanner：词法分析器的全部代码

　　Parser：语法分析器的全部代码

　　Semantics：语义分析计算，应用于在 parser 代 码中以构成“解释器”

　　Errlog：错误处理，信息输出

ui\_main：人机界面+主程序，文本交互consol 或图形界面WinGUI

**词法分析器：**

1. 任务

·滤掉源程序中的注释和无用成分(空格、TAB等)；

·输出记号,供语法分析器使用；

·识别非法输入，并将非法输入作为出错记号提供给语法分析器，以便进行出错处理。

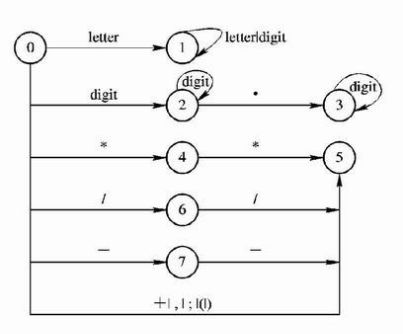
2. 记号设计

·记号类别：规定所有记号的类别（命名常数便于记忆/理解）

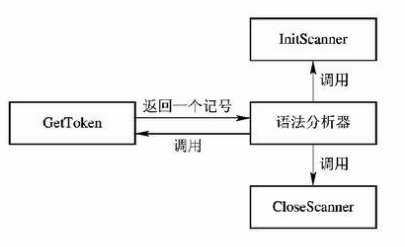
·模式的正规式表示：描述所有类别的记号的模式，将据此构造 DFA

·区分记号的符号表：作用是对按同一模式(ID)识别出来的记号之种类和其他信息进行区别

3. 正规式的DFA

--->

4. 程序框架



(1) 采用“表驱动”模式实现，定义在文件 dfa.c 中。

·将 DFA 的使用抽象为三个接口操作，从而将实现细节封装起来。

·将表驱动型 DFA 的状态转换矩阵当作“稀疏矩阵”，并采用了基于三元组的“压缩”存储策略。

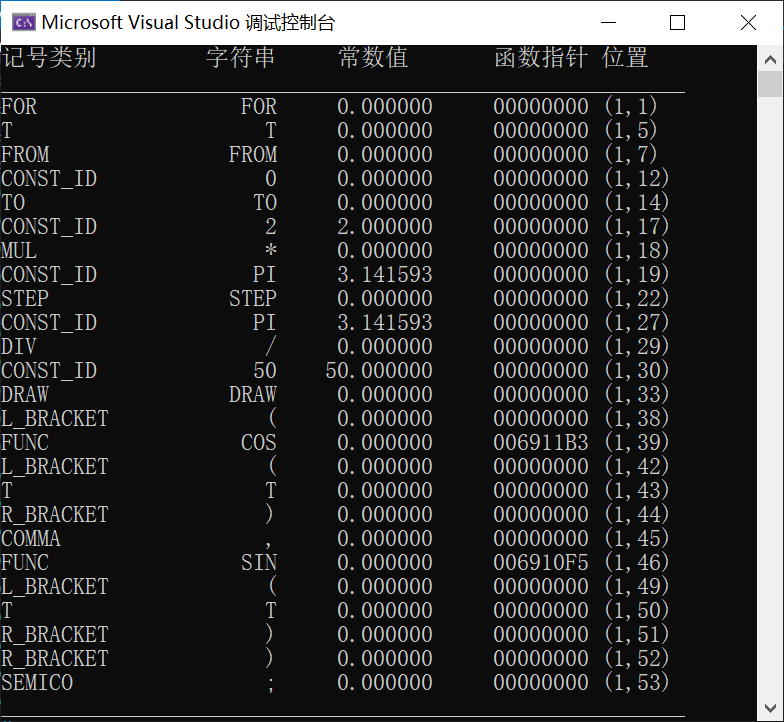
(2) 错误处理策略

·当打开文件失败时，仅通过 InitScanner() 的返回值向调用者报告。

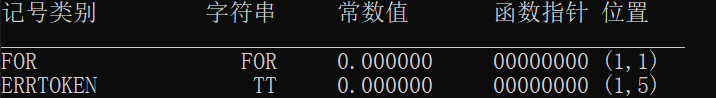
·对于词法分析发现的错误，统一由 GetToken() 返回一个种类为 ERRTOKEN 的记号，而该错误的显示、处理均交由调用者负责。

5. 测试

for t from 0 to 2\*pi step pi/50 draw (cos(t),sin(t));



for tt from 0 to 2\*pi step pi/50 draw (cos(t),sin(t)); --未定义的记号tt



**语法分析器：**

1. 函数绘图语言的文法

采用递归下降子程序方法，要求文法是LL(1)的，不能有二义性、左因子和左递归，同时需要适合于编写递归下降子程序。

2. 表达式的语法树

仅为各类表达式构造语法树，且一个语句分析结束后，该语句对应的全部语法树会被销毁。

3. 递归下降子程序

实质上就是对右部文法符号的展开，遇到终结符直接匹配，遇到非终结符就调用对应的子程序。

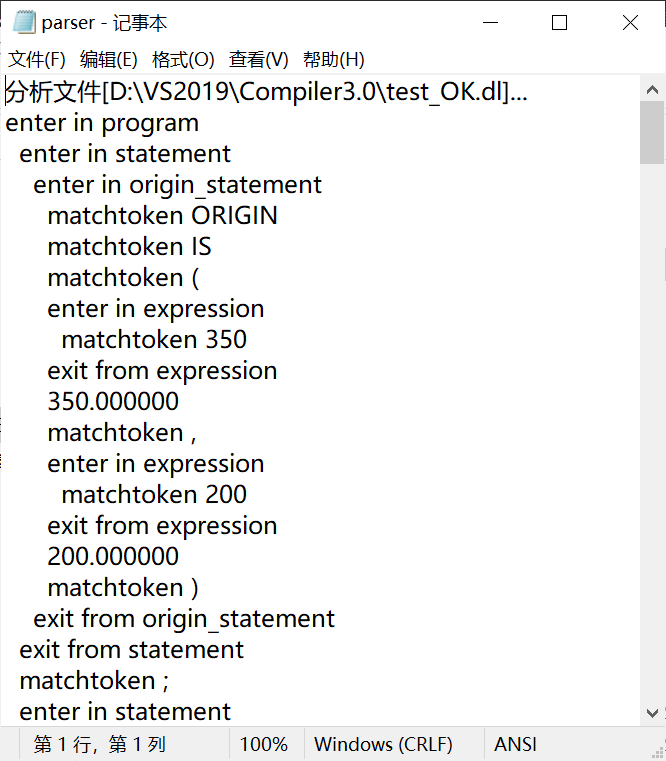
4. 错误处理策略

但凡发现语法错误、词法错误，则显示错误信息之后，立即结束程序。

5. 测试

错误信息输出：写入文件 error.log，也会显示到屏幕上。

其他信息输出：写入文件 parser.log ，不输出到屏幕上。



**语法制导翻译绘制图形：**

1. 绘图语言的语义

语法制导翻译的基本思想是，根据语法分析所得到的语言结构加入相应的语义动作，以实现设计者所希望的目标。对于递归下降子程序，就是在各子程序中的适当位置嵌入相应的语义函数。

绘图语言的语义由两大部分组成:表达式值的计算和图形的绘制。由于在语法分析阶段已经为表达式构造了语法树，因此表达式的计算十分简单，只要从分析树的根节点出发，对分析树进行一次深度优先遍历即可得到表达式的值。

图形绘制相对复杂一些，

(1)从origin、rot 和scale语句中的表达式得到坐标变换所需的信息:

(2) for\_draw 语句根据t的每一个值进行如下处理:

·计算被绘制点的横、纵坐标值:

·根据坐标变换信息进行坐标变换，得到被绘制点的实际坐标值:

·根据点的实际坐标绘图。

2.语义函数设计

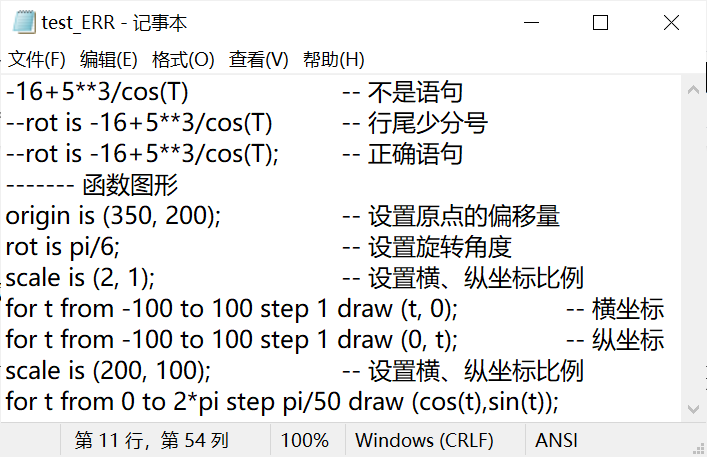
函数绘图语言的源程序是顺序执行的，任一时刻变量T和记录坐标变换信息的内容只需要保存当前最新的值，也就是说任何时刻只需要一套变量。因此可以将它们设计为全程量以减少语义函数之间的参数传递。

3. 递归子程序中语义函数的嵌入

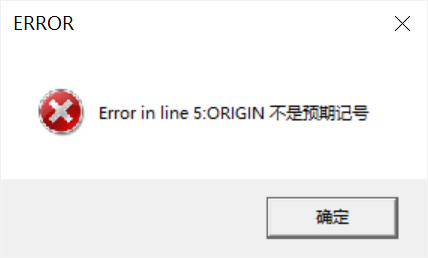
语义可以嵌入在递归子程序中的任何位置，根据语法制导翻译的基本思想，如果希望从某部分语言结构中获取并语义，则相应的语义函数可以紧跟在该结构的语法分析之后。

**测试例程设计与测试结果分析：**

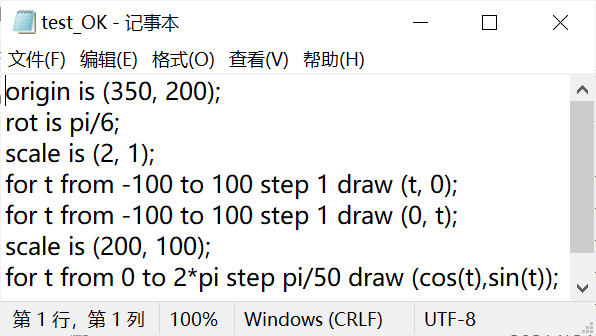
1. 错误测试

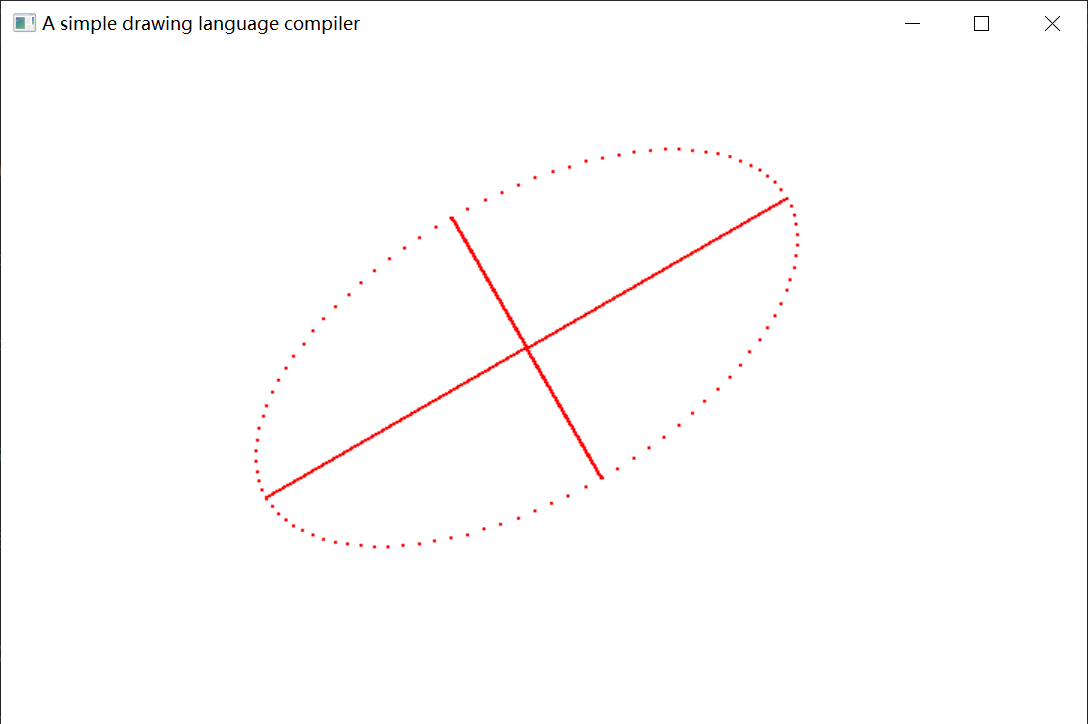


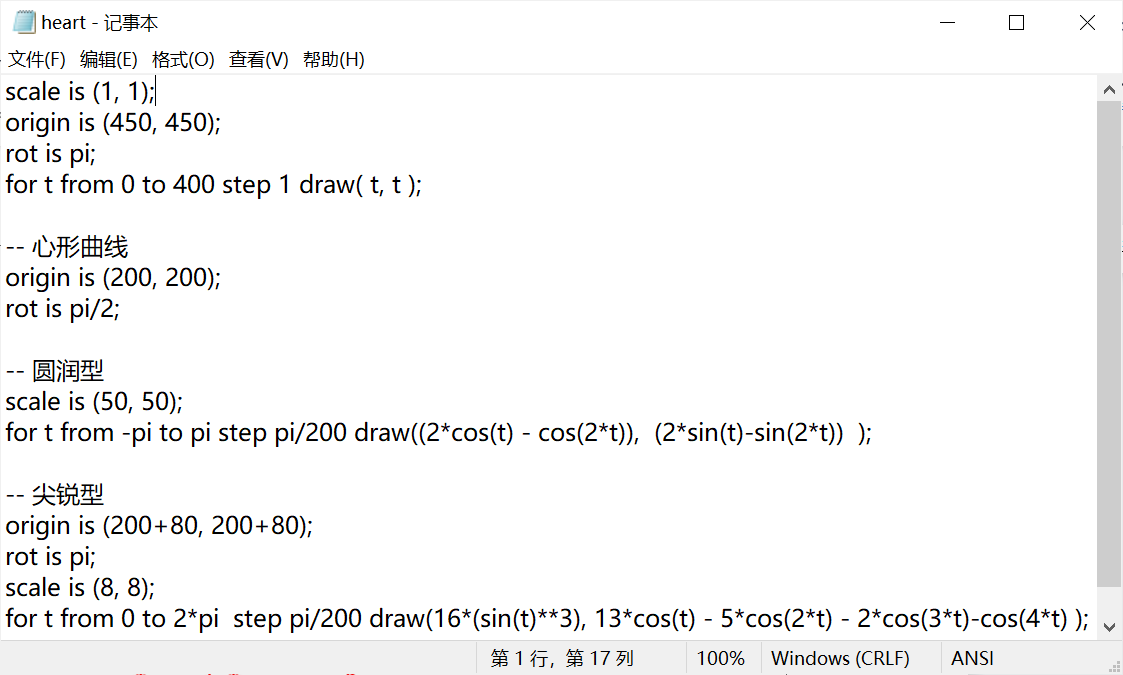
前三行的错误语句分析结果，都是在第五行开头之前报如下错误

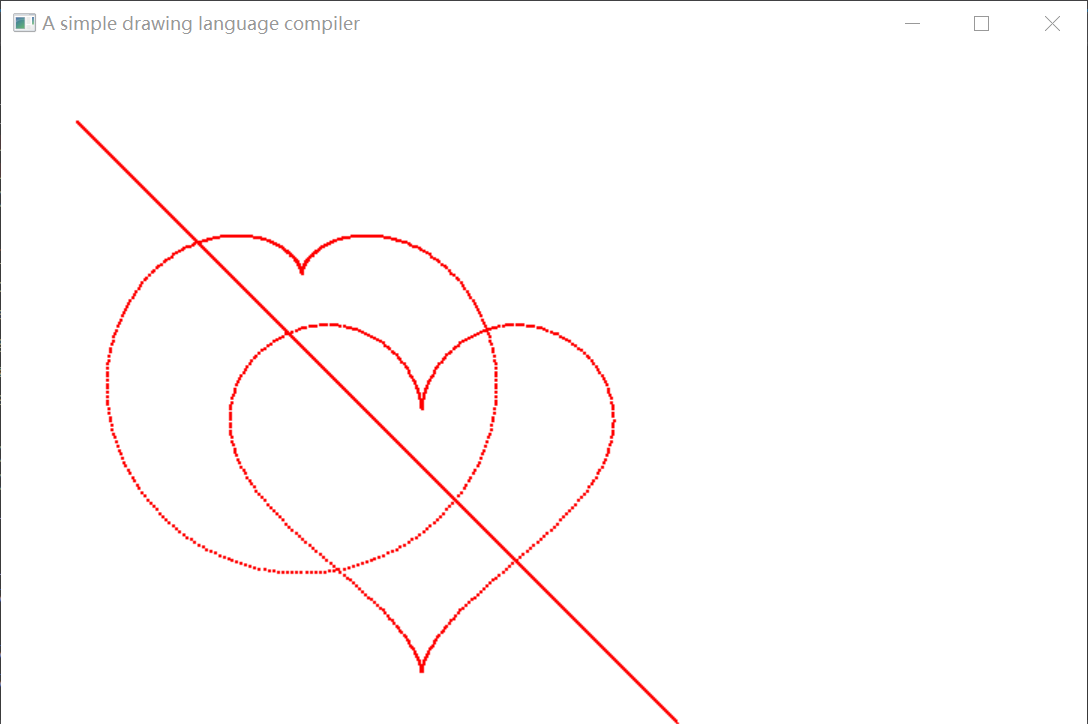


2. 正确测试

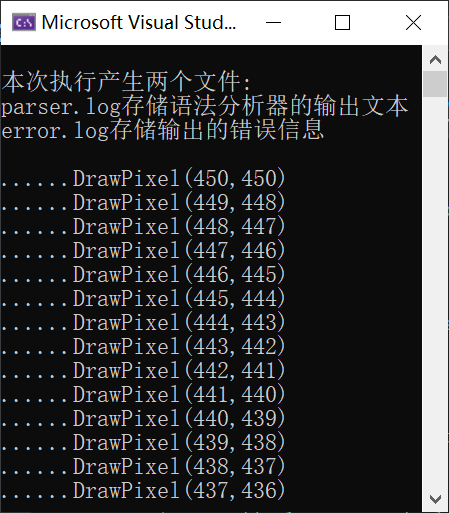


****



****

3. Console版本

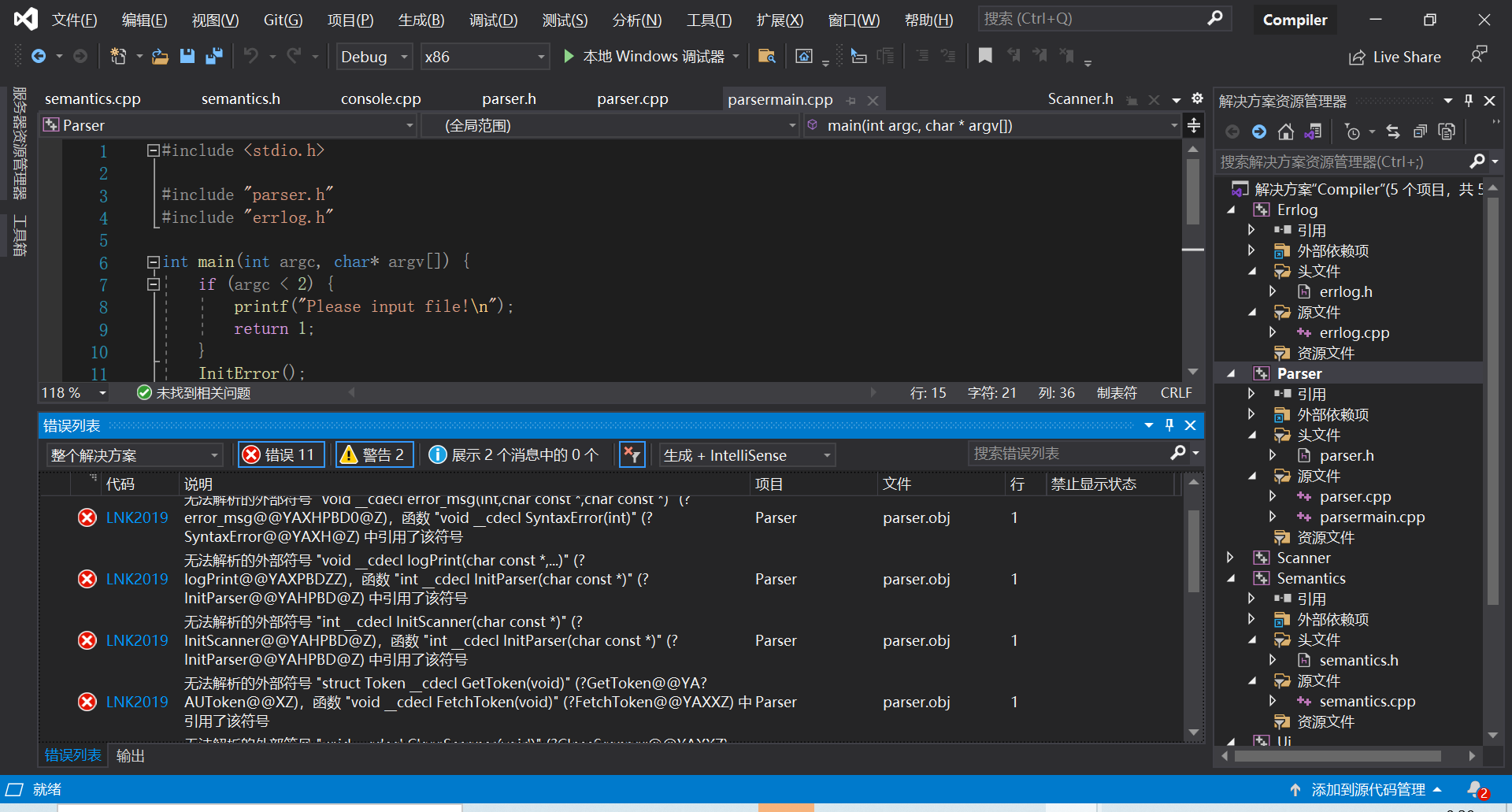


会打印“点阵”。

**四、心得体会**

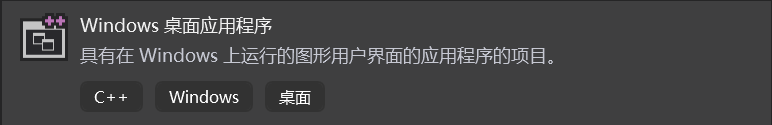
1.源文件组织

需要把所有文件全部放在一个项目中，如果放在同一解决方案下的不同项目中，尽管设置了附加包含目录等，在链接生成.obj文件时还是会报错，如下图。



2.图形化页面

C语言本身所不具备绘图能力，所以实验用到的绘图功能须使用Win32 API，创建Win32 Application。在VS 2019中，如下：



3. 编译时需在编译器的配置中增加宏定义 "\_PARSER\_WITH\_SEMANTICS"，

否则将没有语义计算，图形界面无法执行。