词法分析

为什么用c写：因为c语言是平常联系最多的，无论是学校教的还是自己练的，包括当时的数据结构也是c语言版本的，所以算是比较熟练，就用c语言写了。然后我自己对游戏比较感兴趣，之前也做过两三个游戏的demo都是用c或者c++实现的，感觉还挺有意思的。

看了老师发的资料

makefile：普通的编译肯定是不行，需要用到c++的分离式编译。g++是c++的编译器，如果cpp文件是在一个文件夹中，那可以进入这个文件夹直接g++ 加上源代码的名字就可以编译，但是项目是在不同文件夹下的，不同cpp文件又存在包含其他模块的头文件，此时如果想分离式编译就要先把各个文件夹下的cpp文件先编译为可重定位目标文件，再将这些可重定位目标文件链接为一个可执行文件，每次这样操作的话是非常繁琐的，所以就用makefile将编译链接命令写在一个文件中，再使用使用`mingw32-make`执行文makefile文件。Visual Studio包含了一个组件：VC/VC++，而在VC中，只需要将所有工程文件都包括在project中，VC会自动把makefile写好，用makefile实现分离式编译。

错误情况展示：

错误一，单词错误：记号类型匹配错误，比如将SCALE改成SCALEL

错误二，记号开头错误：比如将参数T改成123T

错误三，表达式错误：比如Draw的参数列表将参数T换为ORIGIN

错误四，保留字匹配错误，语义错误：比如再SCALE IS 后面再加一个ORIGIN

怎么读取文件的：首先在winmain这个函数里将文件名传入，再调用绘图语言解释器，对外接口是parser函数，然后parser函数里面会对词法分析器进行初始化，初始化时会以只读形式打开文件（其实初始化词法分析器和关闭词法分析器就是对应着文件的打开与关闭）

跟踪记号所在的源文件行号，有什么用呢？在语法分析器要用吗？

记录源文件语法出错位置，方便后面的报错窗口对错误类型进行记录。

atof()是C 语言标准库中的一个字符串处理函数，功能是把字符串转换成浮点数

词法分析器：首先设定Token类，然后依据正规式和记号的类型设置查找表。词法分析中，最为重要的就是DFA的实现。首先在gettoken一个记号一个记号的读，首先写一个循环对特殊情况进行处理，用来过滤掉源程序中的空格、TAB、回车等分隔符，文件结束返回空记号，遇到一个字符串尾部为\n就说明要换行了，是使用getchar函数一个字符一个字符的读取文件，getchar函数的实现使用getc来读取文件并将字符转换为大写，并加入到缓冲区数组TokenBuffer，如果遇到空格就说明该记号已经读完了，并将Token类的属性指向他，后期也不用为属性lexme赋值，省去了后期对Token的一小部分赋值的操作。接着判断记号类型，如果是字母打头，就可以采用硬编码的方式，将缓冲区数组送入到JudgeKeyToken对ID，字符常量以及保留字进行识别并返回(其他字符遇到@，#，￥，%，等等就返回ERROTOKEN)。数字只用判断下一个字符是否是数字或者小数点即可。最后判断是否是分隔符或者运算符。

gettoken一直识别到eof位置

注释：如果是-或者/就判断下一个，如果也是的话那就是注释符，整行都不用考虑，这时候就有两种情况，一是这一行不是最后一行，另一种是这一行是最后一行，所以用一个while循环把整行读掉

单字节字符处理函数在**ctype.h**声明。isalnum,isalpha,isspace(是否为空格字符（包括制表符、回车符、换行符等) toupper

语法分析

表达式树结点的类型分为四个。用**匿名联合**使其共用一块内存空间，这样就不用设计四个结点。二元运算的结点（加减乘除，乘方）：类型，左右孩子。函数调用的结点：类型，一个孩子（指向参数），函数指针，说明自己是什么函数。参数T：类型，左值。常量的结点：类型，常量的值。

因为最终构造的是递归下降的语法分析器，要求文法是LL(1)的，所以需要对二义文法进行改造，先改写为非二义的文法，再消除左递归和公共左因子最后转换为产生式与递归子程序有相应关系的形式。

每次分析都是从这一行第一个记号开始，分析到最后分号结束。看看这一行里所有的加号是不是按照scale,rot,orign,for这个四个句型所规定的语法来的。这四个句子，并不存在二义性。而是表达式expression，因为开始的文法没有规定优先级和结合性的定义从而导致了二义性。而且这个程序是左结合和左因子的话，递归调用程序根本没法用，就会在那里无限的递归。

语法分析器的具体思路，从外面整体来看的话，就是打开词法分析器，递归下降分析，关闭词法分析器。递归下降分析中，分为两大类。第一类：四个绘图语句，for,scale,orign,rot。不需要构造语法树，非常的简单。第二类：表达式。需要构造语法树。加减乘除根据书上给的例子。乘方，一元加减。很难。不知道如何下手？后来利用尾递归来实现。

表达式中的运算	结合性	非终结符
(二元)PLUS、MINUS	左结合	Expression
MUL、DIV	左结合	Term
(一元)PLUS、MINUS	右结合	Factor
POWER	右结合	Component
(原子表达式)	无	Atom

Expression 中只能有加、减运算，对于比加、减运算高一级的乘、除运算，需要引入新的非终结符 Term。根据教材中介绍的改写方法，候选项中的两个 Expression 的其中之一要被替换为 Term，由于加、减运算都是左结合的，因此保留运算符左边的 Expression 不变(即递归非终结符在运算符的左边)，将右边的 Expression 替换为 Term，得到新的 Expression 产生式如下：

Expression → Expression PLUS Term
| Expression MINUS Term

Expression和term里必须要有token_tmp，因为matchToken会读取下一个字符。而且按照While循环的内容，这是从下向上生成语法树。有新的加减号，继续将原来的结点作为左结点，与从后向前推到语法树的想法不谋而合。也反应了左结合性。

Matchtoken取下一个字符这一做法很重要，如果当前字符通过递归下降分析正确，就匹配上了，并取下一个记号分析，看看是不是按照我想要的产生式来。

为什么采用树这个数据结构，因为构建语法分析书的时候需要考虑节点的左右孩子等等，而且在语法分析阶段打印语法树信息用到了树的先序遍历，在语义分析阶段求表达式的值用到了树的后序遍历，所以用树这个数据结构。

如何构建语法树的，因为我们考虑了运算符的优先级和结合性，所以在语法分析中会率先识别出优先级最低的运算符即expression这个非终结符对应的符号，将此作为根节点。其次是内部节点，也就是运算符节点是有两个孩子:加PLUS、减MINUS、乘MUL、除DIV、幂运算POWER（有两个特例就是一元加和一元减，比如+5，这个+的左孩子就是NULL，右孩子是5，就将+5转换成5了；还有-5，这个-的左孩子就是0，右孩子就是5，就将-5转换成0-5了。有一个孩子的内部节点 也就是函数节点:函数FUNC。叶节点:常数CONST_ID、参数T。用一个例子来说明吧：-1+2*cos(T)，是先识别出+号作为根节点，然后他有左右孩子，先看左孩子是-号，有左右孩子，左孩子是0右孩子是-1，再看+号的右孩子是2 * cos(T)，识别出乘号，因为它是位于优先级次低的term，左孩子是常量2，右孩子是函数，其有一个指向函数名的指针，还有一个孩子为参数T。

对生成语法树所使用的变参函数不熟悉，后来直接使用书上代码，当黑箱来用

一个**FOR_DRAW**语句中，点的坐标表达式中可能出现**多个参数T**，它们对应不同的树结点，但均对应同一个(T)的存储区。在结点中保存地址，可以实现多次出现的T内容共享并始终保持统一。

语义分析

计算表达式的值：参数是表达式的根,**后续遍历语法树**,根据不同的节点类型计算当前根节点（表达式）的值

但是在语法分析中打印语法树是先序遍历

将语义分析函数嵌入到递归子程序

主程序

winmain应用程序入口点，函数流程：**注册窗口**；**创建窗口**；**显示窗口**；**调用绘图语言解释器读取文件并解释**；**消息循环**，不断从线程的消息队列中取出消息并进行响应；**消息处理**：Windows程序是事件驱动的，对于一个窗口，它的大部分例行维护是由系统维护的。每个窗口都有一个消息处理函数。在消息处理函数中，对传入的消息进行处理。 客户写一个消息处理函数，在窗口建立前，将消息处理函数与窗口关联。这样，每当有消息产生时，就会去调用这个消息处理函数。通常情况下，客户都不会处理全部的消息，而是只处理自己感兴趣的消息，其他的，则送回到系统的缺省消息处理函数中去。

WinMain函数是提供给用户的Windows应用程序入口点，其原型如下：

HINSTANCE　　hInstance，//当前实例句柄

HINSTANCE　　hPrevInstance，//先前实例句柄

LPSTR　　lpCmdLine，//命令行参数

nCmdShow　//显示状态（最大化、最小化、隐藏）

MSG是Windows程序中的结构体。表示消息，四个成员变量含义：第一个成员变量hwnd表示消息所属的窗口；第二个成员变量message指定了消息的标识符。第三、第四个成员变量wParam和lParam，用于指定消息的附加信息。

程序返回值为 0 - PostQuitMessage（） 给出的值

return messages.wParam;

1+...+99的递归

```
public static int hundred(int i){     *//递归实现1+2+3+...+99+100*
    if(i == 100)return 100;
    else return i + hundred(i + 1);
}
```

最难的部分

写语法分析器的时候，非终结符的递归子程序不太会写，factor、component、atom

看了书和网上的代码，因为一元加减和幂运算都是右结合的，它们都用的是递归调用自己，不太明白为什么这样做

还有创建语法树的时候所运用的变参函数不太熟悉，看得书上的代码

其次还有主程序那一部分，因为之前也从来没有接触过用c或者c++画图之类的，所以一开始就比较蒙，不知道从哪入手。后来看到了书上采用的方法是在vc++环境下调用win32api来窗口化画图，觉得还挺有意思的所以去学了下。