运行截图：

命令行参数错误：

```
yamlite -duang sample.yml
Usage: yamlite [option [value]] file
```

正常运行：

```
yamlite -parse sample.yml
valid
```

修改后含有错误的文件：

```
#键和值之间由英文冒号后跟一个空格分割
#字符串
string: "value
#整数
int: 2333
#浮点数
float: 2333.33
#科学计数法
scientific_notation:1.234e-10
#布尔值
bool: tru

#当值为键值对或者数组时，新增一行，缩进增加两个空格
#数组中的每个元素值跟随在“-”和一个空格后面
array:
  - 1
  - 2
  - 3
  #当值为键值对或者数组时，新增一行，缩进增加两个空格
  -
    - 1
    - 2
    - 3
  -
```

输出含有错误的文件的错误列表：

```
yamlite -parse sample.yml
line 8, position 15: expect <">
line 14, position 20: expect one blank sapce
line 16, position 6: Wrong value type!
line 26, position 6: expect exactly 4 blank space!
```

将 yaml 文件内容输出到 json 文件中：

```
yamlite -json sample.yml
sample.json
```

输出的 json 文件的内容：

{"an_identifier_1":"value","string":"value","int":2333,"float":2333.33,"scientific_notation":1.234e-10,"bool":true,"array":[1,2,3,[1,2,3],{"key":"value"}]}

{"an_identifier_1":"value","string":"value","int":2333,"float":2333.33,"scientific_notation":1.234e-10,"bool":true,"array":[1,2,3,[1,2,3],{"key":"value"}]}

思路：

将文件内容输出得到字符串，并将字符串的编码规范统一，然后按行读取，分析，通过判断是否符合格式来决定是存储 token 还是存储 error，存储 token 时通过判断这一行前有几个缩进来决定这个 token 的 level 值，这个 level 值会在语法分析中起到很重要的作用：同一 level 的 token 会处于语法树的同一层级，最主要的函数是 scanner 函数，它最终会返回一个 tokenlist 和一个 errorlist，如果 errorlist 为空，则这个文件是 valid 的。Parser 利用 lexer 的 tokenlist 建语法树，最终得到一颗完整的语法分析树，可以进行转为 json 文件的输出操作。

所做工作：

我实现了 token 类，error 类以及 lexer 类，token 和 error 主要用于存储，lexer 用于进行词法分析，并返回一个 tokenlist 以供 parser 进行语法分析。

另外，我也实现了命令行参数的分析以及命令的执行选择，和将 YAMLite 文件内容输出为 json 文件的功能，这个主要是通过对 parser 所建成的树进行前序遍历，以及对符号的添加等操作来实现的，输出为一个 stirng，然后再输出到文件中。

此外我也负责了大部分的 debug 和代码的修改工作。