- **写一个var类，其可以自适应任何类型变量，是一个动态类型，并实现重载（var.h文件）**

```cpp
#include<conio.h>

#include"var.h"
using namespace std;

int main() {
    var s1=114.51,s2=1000;
    cout<<s1/3+s2<<"\n";
}
```

```
D:\Users\lhk\Downloads\Dynamically_Parser-main\Dynamically_Parser-main\run.exe
1038.17

--------------------------------
Process exited after 0.3301 seconds with return value 0

Press ANY key to exit...
```

- **写一个词法解析器，能从文件中解析每一行代码(tokenize.h)**

```cpp
#include <bits/stdc++.h>
#include<conio.h>
#include"tokenize.h"
#include"var.h"
#include"parse.h"
#include"parsejson.h"
using namespace std;

void run() {
    ifstream fin("test.a66");
    stringstream ss;
    ss << fin.rdbuf();
    cout << ss.str() << endl;
    string s = ss.str(); s += "                    ";
    auto vec = tokenize::tokenize(s);
    for(auto i:vec) {
        cout<<i.value<<"\n";
    }
    //parse::run(vec);
}
int main() {
    run();
    cout << endl;
    cout << "请输入任意键结束....";
    _getch();
    return 0;
}
```

test.a66

文件　编辑　查看

```
def main(){
        ability_66=114*51.4/3;
}
```

```
def main(){
        ability_66=114*51.4/3;
}

def
main
(
)
{
ability_66
=
114
*
51.4
/
3
;
}

请输入任意键结束....
```

- **写一个语法树进行计算，由于有多种运算方式，需要考虑优先级以及括号的问题，我们可以拿语法树+栈的方式进行模拟(parse.h里的Avl_tree)，然后使用run函数进行计算，对于其他的命令，例如输出等，我们都可以将其看作一个运算符，只是运算级的区别，然后按要求返回结果即可。**

```
01  #include <bits/stdc++.h>
02  #include<conio.h>
03  #include"tokenize.h"
04  #include"var.h"
05  #include"parse.h"
06  #include"parsejson.h"
07  using namespace std;
08
09  void run() {
10      ifstream fin("test.a66");
11      stringstream ss;
12      ss << fin.rdbuf();
13  //  cout << ss.str() << endl;
14      string s = ss.str(); s += "                    ";
15      auto vec = tokenize::tokenize(s);
16      parse::run(vec);
17  }
18
19
20
21
22
23
24
```

test.a66

文件 编辑 查看

```
def main(){
        ability_66=114*51.4/3;
        print ability_66;
}
```

D:\Users\lhk\Downloads\Dynamically_Parser-main\Dynamically_Parser-main\run.exe

```
1953.2
请输入任意键结束....
```