abstract syntax tree

关于py3抽象语法树的说明

编译：

makefile已经写好在根目录下，它会自动编译目标文件，归档生成库，名为libast，放在lib目录之下，头文件在include目录之下。编译的同时生成了数个ast数据结构的测试文件，在test目录之下。不要单独编译，否则会失败

要编译成功，需要开启c++17标准的支持，并且使用g++（内部已经指定）。归档使用gnu ar工具。由于使用了unix风格的文件名，windows可能会有错误。

使用：

ast的大部分功能都已经在test中展现过，包括各种运算，控制流，循环，函数定义，最好的办法是直接查看编写的各种测试文件。

使用时只需要include "AST.h"即可。虽然单独包含也没什么问题（写在总头文件里的警告只是个彩蛋而已）

除开单元运算，二元运算等固定的结构，其他可变的结构都采用可变模板处理，应用了大量的泛型编程。在头文件中定义了create宏，这可以方便地创建结构，不需要关心内部到底是原生指针还是智能指针。需要注意的是运行期能否动态地创建结构就需要parser来干了。

问题：

parser需要注意这些问题：

- 如果一个函数的参数有默认值，语法树不会检查是不是默认值都在最后

- 运行效率低
- 你可能需要用宏构建一层新的接口，因为所有结构都是模板写的，不确定能不能实现运行期多态
- 千万不要使用new和原生指针
- 不要手动delete和free
- 潜在的bug和设计缺陷

注释占了很大一部分，支书加油