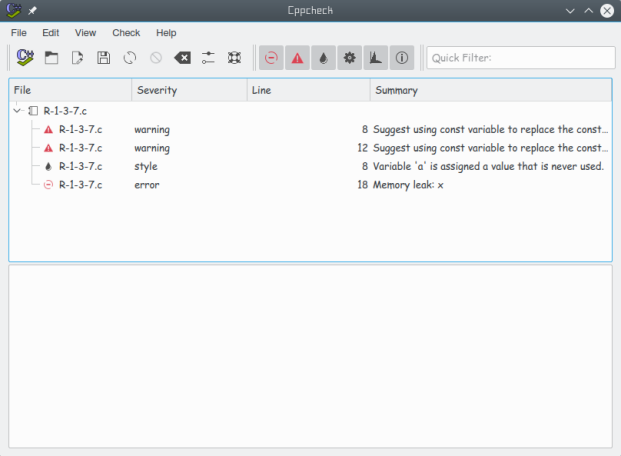
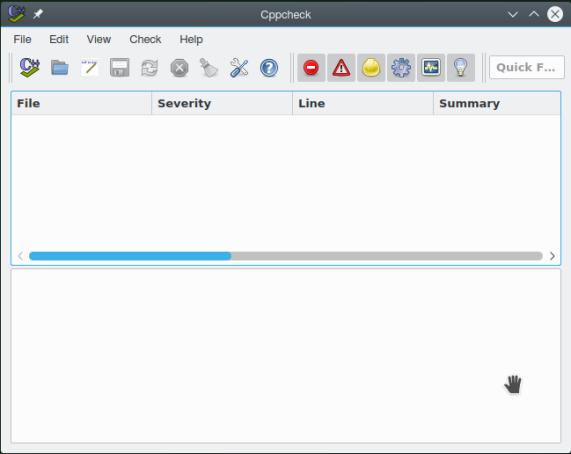
**TooSimple小组算法设计**

GUI

**界面美化**

界面美化主要采用替换图标和适当的UI文件修改，选择替换的图标方案是Linux的KDE桌面经典图标主题Breeze，从一千多个矢量图标里精选二十多个，替换完后界面更加的扁平话，简约的图标使整体简洁大气。(对比图，后者为修改后样式)

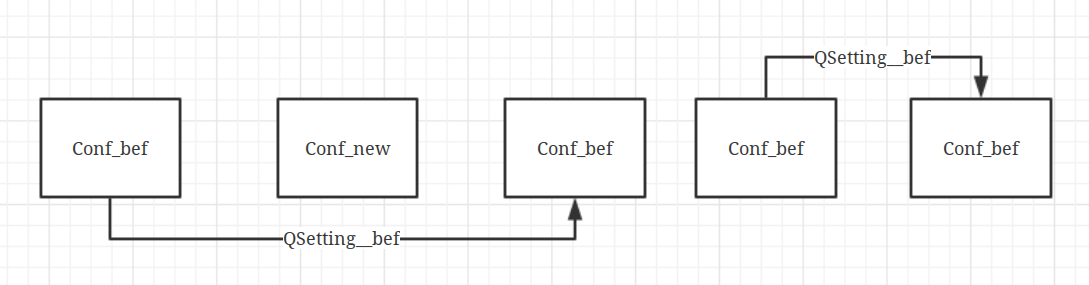


**偏好导出**

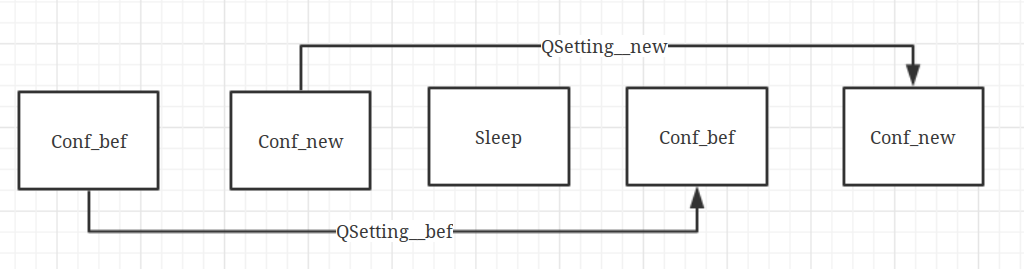
在UI文件中加入相关按钮，同时写链接类，新建的功能函数中主要实现的导出方法是通过调用系统参数获取一个官方文档指定的环境变量（Linux,Mac,Windows不同），从而获取原本配置文件所在的路径，在调用文件对话窗口让用户选择保存路径和导出文件名，从而实现一个从默认配置文件到用户选择的路径和文件的复制操作。

**偏好导入**

由于原软件设计调用的是软件打开时QT自动实现的导入配置文件接口，因此无法达到重写的目的，采用的变通方法是导入文件采用如同导出一般的操作，然后自动重启软件实现新导入偏好的即时应用功能。而有由于软件在关闭时会自动保存导出设置(通过QT的Qsetting接口，无法取消)，如下图：



因此采用如同下图的结构，在新配置覆盖原配置文件后新打开窗口之后再关闭原窗口，来保证功能的完美实现。



规则开发：

自定义规则开发：

**变量在定义的时候建议先初始化**

使用模式匹配的方法，遍历整个代码，找到代码中定义的变量，然后再变量后续代码，找出对变量进行赋值的代码，从中找出赋值的变量，然后判断定义的变量和后续赋值的变量是否统一，如果不统一，那么提示其需要对其进行初始化。

**强制类型转换成整型的时候出现数据溢出的问题**

遍历代码，找到定义整型的代码，然后取出对 Int型变量的赋值，将这个值存为字符串类型，然后进行字符串操作，得到这个值的整数部分，将这个整数和整型变量的取值范围进行比较，如果超出范围，那么就给出溢出的提示信息。

**在布尔表达式中出现赋值则给出提醒**

输入：代码

输出：提示信息

流程：

1、遍历所有的token，若找到while或if，则进行下一步

2、判断while或if后面紧跟的括号中的表达式时候含有赋值号，若有，则给出提示信息。

所要求规则开发：

**A-1-1-6建议用宏或const定义常数。**

输入：代码

输出：提示信息

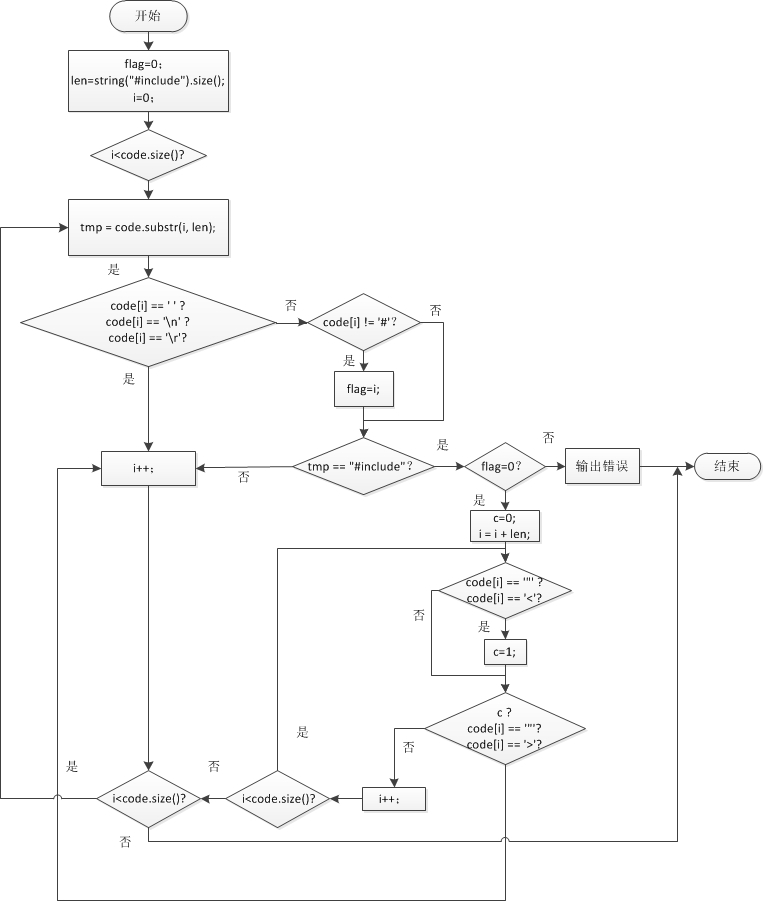
流程：

1、遍历所有的token，令tok指向当前遍历到的token

2、进行模式匹配，调用Token::Match(tok, “%op%|%comp% %num%”)，若该函数返回真，则输出提示信息

**R-1-2-3禁止在头文件前有可执行代码。**

在预处理类中编写检查函数checkCodeBefore(const std::string &code)来实现对#include前是否有代码的检查，并且在read()函数中调用我改写的函数来进行功能的实现。函数的参数是经过read()函数处理过的无注释的代码所生成的字符流，从字符流起始位置开始查找“#include”的位置并记录，若“#include”的位置不在0下标处则报错并将函数直接返回；若当前“#include”函数位置在0下标处，则跳过当前行，对后边的代码继续检查。



**R-1-2-5 逻辑判别表达式中的运算项必须要使用括号。**

采用字符串匹配方法，若是一个串匹配上”运算符 变量或数字或字符串 操作符”或者”操作符 变量或数字或字符串 运算符”则可以判断其违反了规则R1-2-5

**R-1-3-7 动态分配的指针变量定义时如未被分配空间必须初始化为NULL**

采用字符串匹配的方法，由于对形如“int \* a = ( int \* ) malloc (sizeof(int));”这样的语句，在规范化阶段会自动变为”int \* a ; a = ( int \* ) malloc ( sizeof ( int ) ) ;”这样两句话，因此若一个串匹配上了”类型 \* 变量名 ;”，并且其后一句匹配上了”变量名 = ( 类型 \* ) malloc”，或者“变量名 = NULL”则说明其合法，若后一句未匹配上，则说明其非法，违反了规则R1-3-7。

**R-1-3-8 动态分配的指针变量第一次使用前必须进行是否为NULL的判别。**

遍历整个代码，找到在代码中定义的指针变量，然后再继续遍历后续代码，找到代码中进行判断NULL的部分，然后找出判空的指针变量，将这个变量和前面代码中定义的变量进行比较，判断两者是否相等，如果相等，那么就判断判空了，否则报出提示信息。

**R-1-6-16 禁止使用已被释放了的内存空间。**

找到代码中free（）的地方，找出free的指针变量，然后在free之后的代码中进行查询，如果前面释放的指针变量在后续代码中进行了调用，那么我们判断被释放的内存空间被使用了，然后报出提示信息。

**R-1-6-17 被free的指针必须指向最初malloc、calloc分配的地址。**

遍历整个代码，找到free（）的部分，然后找出free的指针，判断在前面的代码中是否对指针进行了自加，自减，赋值等操作，如果进行了这些操作，那么就判断被free的指针并没有指向最初分配的地址，从而给出提示信息。

**R-1-3-3禁止对参数指针进行赋值。**

输入：代码

输出：错误信息

流程：

1、取得一个functionScope。

2、对于当前的functionScope，获得对应函数的参数列表。

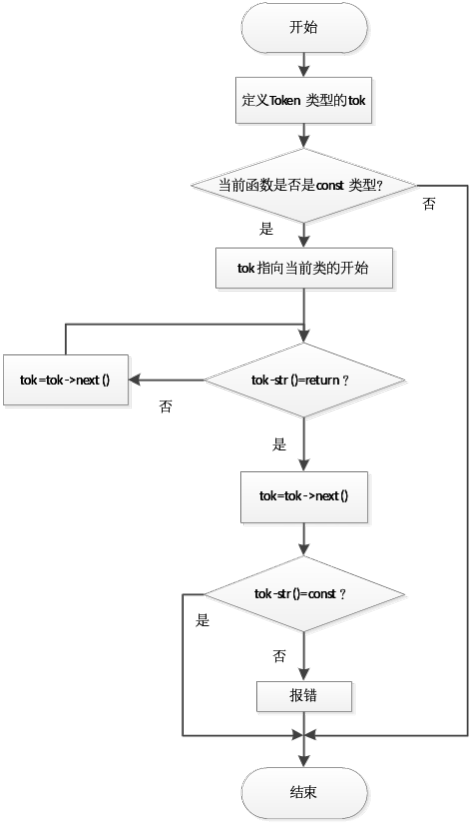
3、将参数列表中的指针变量名存入set中。

4、遍历scope中的所有的token，如果该token在上述的set中且被赋值，则报错。

5、如果还有下一个functionScope，则取下一个functionScope，转2。否则结束。

**R-2-7-2定义为const的成员函数禁止返回非const的指针或引用。**

在checkclass类的checkConst()函数中添加代码，该函数中有一个可以从头到尾遍历一次成员函数的循环，在该循环中添加对成员函数类型的判断，若为const类型，则开始在函数中寻找return的类型，若同为const类型则跳过，开始检验下一个函数，若return类型不符，则报错并开始检验下一个函数。



**R-2-7-3禁止可导致非资源性对象数据被外部修改的成员函数返回。**

输入：代码

输出：错误信息

流程：

1、遍历所有的class scope

2、令集合st为空

3、遍历当前class scope中的所有的成员变量，如果有非公有的成员变量则将其放入集合st中

4、令集合def为空

5、遍历当前class scope中所有的函数，如果该函数域中有新定义的变量，则将该变量放入集合def中，如果该函数的返回类型是指针或引用类型，且返回值中包含不在集合def中而在集合st中的变量则输出错误信息。

**R-2-8-3异常抛出的对象必须使用引用方式捕获。**

输入：代码

输出：错误信息

流程：

1、遍历所有的token，若找到了catch，则进行下一步

2、判断catch中的变量类型是否是对象，若是对象，则进行下一步

3、判断该对象是否是引用类型，如果不是，则输出错误信息

**R-2-8-5禁止显式直接抛出NULL。**

输入：代码

输出：错误信息

流程：

1、遍历所有的token，令tok指向当前遍历到的token

2、进行模式匹配，调用Token::Match(tok, “throw 0”)，若该函数返回真，则输出错误信息。