【注意:】

1、 除明确要求外，已学过的知识中，不允许使用 goto、不允许使用全局变量

2、 本作业仅要求 VS2022 编译通过即可（“0 errors, 0 warnings”）

3、 允许使用 string，允许使用 vector（新要求，具体使用方法自学），不允许使用其它 stl 容器

综合题 4：完成一套读配置文件的工具集

【背景描述:】

在 Windows 和 Linux 操作系统中，很多应用程序都有相应的配置文件，用来设定程序运行过程中的各个选项，配置文件的结构说明如下：

```
;这是某程序的正常配置文件
;2024.11.18 修订

[VideoProperties]
Title=属性设置
Title_V=10

[SpecialEffect]
Title=特效
EffectBlock=12.3 #版本
ZoomBlock=

[FaceTrack]
   Title =   人脸追踪
FaceTrackingBlock=y

# FaceTrack=3
```

```
;这是某程序的简单配置文件

Title_V=10
EffectBlock=12.3 #版本
ZoomBlock=
   Title =   人脸追踪
FaceTrackingBlock=y
//FaceTrack=3
```

```
;这是某程序的混合配置文件

Title_V=10
EffectBlock=12.3 //版本

[FaceTrack]
ZoomBlock=
   Title =   人脸追踪
FaceTrackingBlock=y
#FaceTrack=3
```

★ 配置文件为文本文件，支持 Windows/Linux 格式，限定每行最长 1024 字节，超过则报错

★ 如果某行出现;或#或//(均为半角)，则表示该符号出现至本行尾部均为注释(左侧红色)，不需要符合语法要求，不计入有效内容中

例 1：某行内容"#xxxx yyyzzz"，则整行均为注释行

例 2：某行内容"name=aa #xxxx yyyzzz"，则"name=aa "是有效内容

★ 配置文件分为若干组，每组用[***]表示组名，组名各不相同

★ 每组有若干项，每项的基本格式是“项目名=值”，同组的项目名不相同，不同组可能相同

- 每个项目一行，不允许多项目一行
- =称为分隔符，常用的配置文件分隔符有=和空格(含 tab)两种，为了方便，后续出现=的位置均可以理解为分隔符
- 项目名为字符串，由英文/中文/数字/符号等组成，均为图形字符
- =后面的内容称为项目值，形式为一行，中间允许空格

★ 项目值的可能取值有：整数、浮点数、单字符、字符串、IP 地址、空

- 字符串可能为英文/中文/数字/符号等
- 字符串不建议包含“;#=[]”等特殊含义的半角字符
- 项目名及值的前后允许有空格、tab 等，不包含在内，也不算错误(左侧例子中[FaceTrack]仍为 Title=人脸追踪)

★ 某些配置文件，可能只有项目名，没有组名，下文中称为简单配置文件；有组名/项目名的则称为正常配置文件；如果文件一开始有若干配置项，再跟组名，则称为混合配置文件

★ 其他

- 定义一行的统一处理顺序：取出一行后，先截断;或#或//开始的注释，再去除前后空格/tab，剩下为有效内容
- 有效内容第一个是[，最后一个是]，就认为是组名，否则不是
- 组名内部允许带空格，但忽略前后空格
  例：某行“ [abc]def] # 测试”，则组名=“abc]def”
- 不含=的项名直接忽略即可（不读取，也不必报错）

【文件处理逻辑：】

1、 文件为文本文件，支持 Windows/Linux 两种格式
2、 每行含行结束符不超过 1024 字节，否则报"非法格式的配置文件，第[xx]行超过最大长度 1024."，不再进行后续操作
3、 最后一行允许不含行结束符

【行处理逻辑：】

1、 取出一行后，先截断;或#或//开始的注释，再去除前后空格/tab，剩下为有效内容
2、 有效内容为空则直接忽略该行
3、 有效内容的第一个字符是[，最后一个字符是]，则认为是组名，否则认为是项，均称为有效行
4、 如果该行是项，则在有效内容中查找分隔符（如果有多个，则认为左起第一个是分隔符，其它是值）
5、 将项以分隔符做截断，左侧是项名，右侧是项值，将项名和项值再做一次去除前后空格/tab 的操作，分别称为项目和项值
6、 项值允许含空格、转义符、单双引号等，为了简化，均不做特殊处理，都当做普通字符
7、 从项值中取整数、浮点数、单字符、字符串、IP 地址等各种数据类型时，统一的处理方式为将值放入 istringstream 中，再>>方式提取第一个有效值，fail()为 0 时取到的数据可信，否则不可信
8、 如果某行的有效内容非空，不是组，也没有分隔符，则也当做项，但仅能被 get_item_all 按原始方式全部读取时被读取
9、 约定一个配置文件只支持一种分隔符方式，在初始化时确定，如果有其它分隔符，可按原始内容读出后自行进行后续处理

例 1：某行" [abc def] #test" => 组名"[abc def]"
例 2：某行" name = 张三 #姓名"
  项名=>"name"
  项值=>"张三"，按 cstring/string 均取到"张三"，char 取到半个汉字，int/double/ip 非法
例 3：某行" name = \"\t 张三 李四\" #姓名"，默认分隔符为=
  项名=>"name"
  项值=>"\"\t 张三 李四\""，按 cstring/string 均取到"\"\t 张三"，char 取到"\"，其余非法
例 4：某行" name = \"\t 张三 李四\" #姓名"，指定分隔符为空格
  项名=>"name"
  项值=>"= \"\t 张三 李四\""，按 char/cstring/string 均取到"="，其余非法
例 5：某行" name \"\t 张三 李四\" #姓名"，默认分隔符是=，则该行无法匹配项名和项值，仅能被 get_item_all 整体读取
【注】：前后的红色"不是行的有效内容，仅为了方便分辨是否有空格

【工具函数集的定义】

class cfgfile_read_tools 的定义放在 class_crt.h 中，各成员函数的说明如下：

★ cfgfile_read_tools(const char* const cfgname,
  const enum BREAK_CTYPE bctype = BREAK_CTYPE::Equal)
使用说明：构造函数，指定要读取的配置文件名及分隔符的形式

- cfgname 指定配置文件名，可带绝对路径/相对路径
- 分隔符有=和空格（含 tab）两种形式，具体见 class cft.h 中的定义，默认为=
- 还有一个重载函数，适用 string 形式的文件名，其余同

★ ~cfgfile_read_tools()
使用说明：析构函数，按需放入需要的内容

★ bool is_read_succeeded()
使用说明：判断配置文件是否打开成功

★ int get_all_group(vector <string>& ret)
使用说明：返回配置文件中的所有组，放在 vector 中
- 成功返回组的数量，失败返回 0（返回 0 时 vector 为空）
- 如果有多个 group 相同则全部返回（即只要取到所有[**]的行，不考虑是否重复）
- 对于简单配置文件，返回一个组，组名为""，返回值为 1
- 对于混合配置文件，第一个组是""，后续是有组名的组

★ int get_all_item(const char* const group_name,
vector <string>& ret,
const bool is_case_sensitive = true)
使用说明：返回 group_name 组中的所有项，放在 vector 中
- is_case_sensitive 为 true 表示 group_name 大小写敏感，false 表示大小写不敏感，默认为 false
- 成功返回该组中项的数量，无任何项则返回 0（返回 0 时 vector 为空）
- 返回的是该组的有效行的全部内容的完整字符串形式（不考虑值类型，供后续自行处理；全部有效内容指去除项目名之前/值之后的空格、tab 等，下同），不考虑是否含分隔符，也不考虑项是否重复
- 对于简单配置文件，group_name 指定为""（如果指定为 nullprt/NULL 则直接返回 0）
- 还有一个重载函数，group_name 为 string 类型（即形参为 const string &group_name），其余同

★ int item_get_raw(const char* const group_name,
const char* const item_name,
string &ret,
const bool group_is_case_sensitive = true,
const bool item_is_case_sensitive = true)
使用说明：返回 group_name 组中的 item_name 项的内容，放在 string 中
- group_ignore_lower_upper_case 为 true 表示 group_name 大小写敏感，false 表示大小写不敏感，默认为 true（后续均相同，不再重复说明）
- item_ignore_lower_upper_case 为 true 表示 item_name 大小写敏感，false 表示大小写不敏感，默认为 true（后续均相同，不再重复说明）
- group_name 的处理同 get_all_item（后续均相同，不再重复说明）
- 成功返回 1，失败返回 0（返回 0 时 ret 的值不可信）
- 返回形式是“项目名 = 值”的有效行的全部内容
- 如果 item 值相同的有多项，取第一项
- 还有一个重载函数，group_name/item_name 为 string 类型，其余同

★ int item_get_null(const char* const group_name,
const char* const item_name,
const bool group_is_case_sensitive = true,
const bool item_is_case_sensitive = true)
使用说明：判断 group_name 组中 item_name 项是否存在
- 该项存在返回 1，不存在返回 0
- 仅判断该项是否存在，对是否有值、值的具体内容不关心
例："name=" / "name=Y" / "name=张三" / "name=123" 均返回 1
- 还有一个重载函数，group_name/item_name 为 string 类型，其余同

★ int item_get_char(const char* const group_name,
const char* const item_name,
char &value,
const char *const choice_set = nullptr,
const char def_value = DEFAULT_CHAR_VALUE,
const bool group_is_case_sensitive = true,
const bool item_is_case_sensitive = true)

使用说明：按 char 类型读取 group_name 组中的 item_name 项的内容

- 返回值放在 value 中
- choice_set 指定合法的取值（例："YyNn"表示只有大小写 Y/N 是合法的，默认 nullptr 表示任意字符均合法）
- def_value 表示当取值不合法或不存在时，返回默认值，有两种情况
  - def_value 是 DEFAULT_CHAR_VALUE，则不合法/不存在时返回 0
  - def_value 不是 DEFAULT_CHAR_VALUE，则不合法/不存在时置 def_value 并返回 1
  - 具体请参考 test_readcfg 中的测试用例
- 取到合法值/默认值返回 1，否则返回 0（返回 0 时 value 的值不可信）
- 还有一个重载函数，group_name/item_name 为 string 类型，其余同

★ int item_get_int(const char* const group_name,
const char* const item_name,
int &value,
const int min_value = INT_MIN,
const int max_value = INT_MAX,
const int def_value = DEFAULT_INT_VALUE,
const bool group_is_case_sensitive = true,
const bool item_is_case_sensitive = true)

使用说明：按 int 类型读取 group_name 组中的 item_name 项的内容

- 返回值放在 value 中
- [min_value..max_value]指定了合法取值的范围（例：[0~100]表示成绩范围，默认为全部 int 值）
- def_value 表示当取值不合法或不存在时，返回默认值，有两种情况
  - def_value 是 DEFAULT_INT_VALUE，则不合法/不存在时返回 0
  - def_value 不是 DEFAULT_INT_VALUE，则不合法/不存在时置 def_value 并返回 1
  - 具体请参考 test_readcfg 中的测试用例
- 取到合法值/默认值返回 1，否则返回 0（返回 0 时 value 的值不可信）
- 还有一个重载函数，group_name/item_name 为 string 类型，其余同

★ int item_get_double(const char* const group_name,
const char* const item_name,
double &value,
const int min_value = DBL_MIN,
const int max_value = DBL_MAX,
const int def_value = DEFAULT_DOUBLE_VALUE,
const bool group_is_case_sensitive = true,
const bool item_is_case_sensitive = true)

使用说明：按 double 类型读取 group_name 组中的 item_name 项的内容

- 返回值放在 value 中
- [min_value..max_value]指定了合法取值的范围（例：[3.8~4.3]表示电压范围，默认为全部 double 值）

- def_value 表示当取值不合法或不存在时，返回默认值，有两种情况
  - def_value 是 DEFAULT_DOUBLE_VALUE，则不合法/不存在时返回 0
  - def_value 不是 DEFAULT_DOUBLE_VALUE，则不合法/不存在时置 def_value 并返回 1
  - 具体请参考 test_readcfg 中的测试用例
- 取到合法值/默认值返回 1，否则返回 0（返回 0 时 value 的值不可信）
- 还有一个重载函数，group_name/item_name 为 string 类型，其余同

★ int item_get_cstring(const char* const group_name,
const char* const item_name,
char* const value,
const int str_maxlen,
const char* const def_value = DEFAULT_CSTRING_VALUE,
const bool group_is_case_sensitive = true,
const bool item_is_case_sensitive = true)

使用说明：按 char 数组/指针类型读取 group_name 组中的 item_name 项的内容

- 返回值放在 value 中
- 调用者需要保证 value 有足够的空间（char 数组或者 char*但已动态申请足够的空间），str_maxlen 指定了传入的 char 数组/指针含尾零能存放的最大字符数（含尾零），最长不超过 MAX_STRLEN（即使给的空间再大，最多也就读 MAX_STRLEN-1 个字节）
  - **注意：**测试用例中有一个运行错误的测试项!!!
- def_value 表示当取值不合法或不存在时，返回默认值，有两种情况
  - def_value 是 DEFAULT_CSTRING_VALUE，则不合法/不存在时返回 0
  - def_value 不是 DEFAULT_CSTRING_VALUE，则不合法/不存在时置 def_value 并返回 1
  - 具体请参考 test_readcfg 中的测试用例
- 取到合法值/默认值返回 1，否则返回 0（返回 0 时 value 的值不可信）
- 还有一个重载函数，group_name/item_name 为 string 类型，其余同

★ int item_get_string(const char* const group_name,
const char* const item_name,
string& value,
const string& def_value = DEFAULT_STRING_VALUE,
const bool group_is_case_sensitive = true,
const bool item_is_case_sensitive = true)

使用说明：按 string 类型读取 group_name 组中的 item_name 项的内容

- 返回值放在 value 中
- def_value 表示当取值不合法或不存在时，返回默认值，有两种情况
  - def_value 是 DEFAULT_STRING_VALUE，则不合法/不存在时返回 0
  - def_value 不是 DEFAULT_STRING_VALUE，则不合法/不存在时置 def_value 并返回 1
  - 具体请参考 test_readcfg 中的测试用例
- 取到合法值/默认值返回 1，否则返回 0（返回 0 时 value 的值不可信）
- 还有一个重载函数，group_name/item_name 为 string 类型，其余同

★ int item_get_ipaddr(const char* const group_name,
const char* const item_name,
unsigned int& value,
const unsigned int& def_value = DEFAULT_IPADDR_VALUE,
const bool group_is_case_sensitive = true,
const bool item_is_case_sensitive = true)

使用说明：按 IP 地址类型读取 group_name 组中的 item_name 项的内容

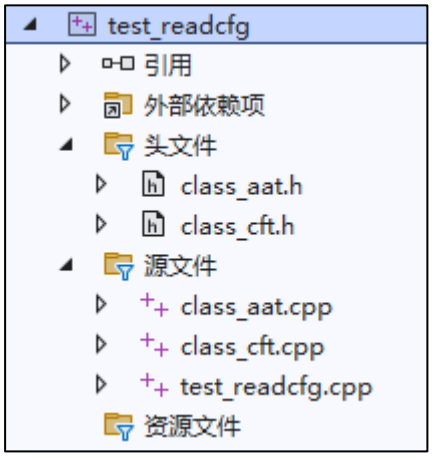
- 返回值放在 value 中（是点分十进制的整型表示，不是"1.2.3.4"形式的字符串）
- def_value 表示当取值不合法或不存在时，返回默认值，有两种情况
  - def_value 是 DEFAULT_IPADDR_VALUE，则不合法/不存在时返回 0
  - def_value 不是 DEFAULT_IPADDR_VALUE，则不合法/不存在时置 def_value 并返回 1
  - 具体请参考 test_readcfg 中的测试用例
- 取到合法值/默认值返回 1，否则返回 0（返回 0 时 value 的值不可信）
- 还有一个重载函数，group_name/item_name 为 string 类型，其余同

【实现要求：】

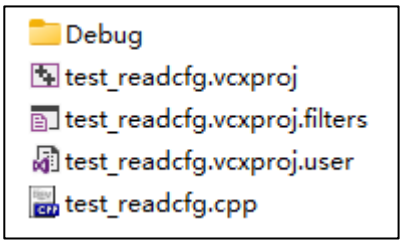
1、在 BigHW 中新建项目 test_readcfg（注意：下划线）

a) 附件中的 test_readcfg.cpp 放入 test_readcfg 中
b) 附件中的 class_cft.h 放入 include 中
c) 附件中的 class_cft.cpp 放入 common 中
d) 将 class config_file_tools 的定义和实现补充完整
e) 通过 test_readcfg.cpp 的测试（有宏定义 TEST_FOR_FIXED_FILE，0/1 均需要通过）

2、项目和对应文件夹的文件说明（项目目录中**仅允许**测试文件及配置文件样例存在）



1、 项目中允许加入你自己的工具函数，但实际文件应该位于 common/include 的公共目录下
2、 文件夹中还允许包含测试用的配置文件



3、鼓励合理拆分源程序文件、合理划分函数、合理共用公共函数等
4、修改 common/include 中的内容后，要保证之前的所有项目能编译通过并运行正确
5、提供 test_readcfg.exe 供参考（固定读和自由读分别对应宏定义 TEST_FOR_FIXED_FILE）
6、附件中提供了若干配置文件供参考，有 Windows/Linux 两种格式

【提交要求（仔细阅读，当心 0 分!!!）：】

1、提交作业前，先做好完整备份
2、之前大作业的lib记得删掉（文件夹及项目都要删除，**之前犯过错的同学记得更正!!!**）
3、要保证BigHW的所有项目都能编译通过
4、按之前的BigHW提交要求，整个BigHW目录压缩成BigHW.rar，再按网页要求改名后提交

【编译器要求：】

仅 VS2022 通过即可

【作业要求：】

1、12 月 1 日前网上提交本次作业
2、每题所占平时成绩的具体分值见网页
3、超过截止时间提交作业则不得分