【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 类，不允许使用其它 stl 容器

综合题 5-1：完成一套配置文件读写的工具集

【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：

```
;这是某程序的配置文件
;2022.05.19 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
```

★ 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

★ 值的可能取值有：整数、浮点数、单字符、字符串、空

- 字符串可能为字母、数字、中文、符号等
- 字符串不含空格，tab 键等不可显示字符
- 字符串不含 TAB、;、#、″、’、{}、[]、()、=等特殊含义字符(均为半角字符)
- 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)

★ 如果某行出现;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，也不被读取

★ 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

```
;这是某程序的配置文件
;2022.05.19 修订

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

★ 其他

- 组名/项目名/值均可能有中文
- 定义一行的统一处理顺序：取出一行后，先截断;及#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=的项名直接忽略不处理即可（不必报错）

【工具函数集的定义(C++方式)】

class cft(cfg_file_tools)的定义放在class_cft.h中,各成员函数的说明如下(假设cft fcfg):

★ void open(const char* cfgname, int opt = OPEN_OPT_RDONLY)

使用说明:

fcfg.open("test.cfg", OPEN_OPT_RDONLY): 表示用只读方式打开文件"test.cfg"

fcfg.open("test.cfg") : 同上

fcfg.open("test.cfg", OPEN_OPT_RDWR) : 表示用读写方式打开文件"test.cfg"

- 读写方式定义如下

```
#define OPEN_OPT_RDONLY     0   //以只读方式打开（打不开则返回失败）
#define OPEN_OPT_RDWR       1   //以读写方式打开（打不开文件则创建）
```

  - ◆ 读写方式: OPEN_OPT_RDONLY 表示只读，这种打开方式适用于 item_get_value 函数
  - ◆ 读写方式:OPEN_OPT_RDWR 表示读写,这种打开方式适用于所有函数,若文件不存在,要创建新文件
- 两参构造函数 CFT(const char* cfgname, int opt = OPEN_OPT_RDONLY) 功能同 open
- 测试用例开始必须调用 open/两参构造函数之一，才能正确进行后续的操作
- 因为构造函数不能有返回值，为保持一致，本函数也定义为 void，统一用 is_open 函数来判断打开配置文件正确/错误
- open 及两参构造均有 string 类型重载，功能相同，此处略

★ void close()

使用说明:

在测试用例的最后可以调用 fcfg.close()来关闭配置文件

- 析构函数~CFT()的功能同 close()，要保证用户显式 close 后析构不出错

★ void set_comment_switch(CommentSwitch flag)

★ CommentSwitch get_comment_switch()

使用说明:

用于设置是否需要过滤注释语句/取当前设置值

- CommentSwitch::on 表示要过滤，off 表示不过滤
- #和;(均为半角)表示单行注释，即从该字符开始到行尾均为注释

★ void set_insert_position(InsertPosition pos)

★ InsertPosition get_insert_position()

使用说明:

用于设置是否新增项的插入位置/取当前设置值

- InsertPosition 共三项，具体可参考源文件中的注释

★ int group_add(const char *group_name)

使用说明:

在测试用例中调用 fcfg.group_add("test");，则表示在配置文件的加入[test]组，组中暂时无内容

- 加入的组放在文件的最后
- 增加成功返回 1，否则返回 0
- 如果[test]组已存在，则不能重复增加，直接返回 0 即可
- 如果手工编辑加出两个及以上的[test]组，再调用 group_add，也是直接返回 0 即可

★ int group_del(const char *group_name)

使用说明：

在测试用例中调用 fcfg.group_del("test");，则表示在配置文件中删除[test]组及该组下存在的全部项

- 删除从本组的组名所在行开始，到下一个组名的所在行之间的所有内容，如果是最后一个组则删除到或文件结束处
- 如果[test]组不存在，直接返回 0 即可
- 如果[test]组重复存在（例如：手工修改使两组同名），则要删除所有同名组并返回 2
- 删除成功返回 n(n>0)，表示删除了 n 个[test]组，否则返回 0

★ int item_add(const char *group_name, const char *item_name, const int item_value)

（共 6 个 item_add 重载函数，此处略，具体请看 class_cft.h）

使用说明：

1、假设在测试用例中有：

```
int i = 12345;
fcfg.item_add("test", "起始值", i);
```

则表示在配置文件的[test]组的指定位置处加入 "起始值=12345" 项

2、假设在测试用例中有：

```
double d = 123.45;
fcfg.item_add("test", "起始值", d);
```

则表示在配置文件的[test]组的指定位置处加入 "起始值=123.45" 项

3、假设在测试用例中有：

```
char *s="好日子";
fcfg.item_add("test", "起始值", s);
```

则表示在配置文件的[test]组的指定位置处加入 "起始值=好日子" 项

4、假设在测试用例中有：

```
string s="好日子";
fcfg.item_add("test", "起始值", s);
```

则表示在配置文件的[test]组的指定位置处加入 "起始值=好日子" 项

5、假设在测试用例中有：

```
char c = 'Y';
fcfg.item_add("test", "起始值", c);
```

则表示在配置文件的[test]组的指定位置处加入 "起始值=Y" 项

6、假设在测试用例中有：

```
fcfg.item_add("test", "起始值");
```

则表示在配置文件的[test]组的指定位置处加入 "起始值=" 项（空项）

- **注：“指定位置”** 的定义在 enum class InsertPosition 中，共三种形式
- 增加成功返回 1，否则返回 0
- 如果[test]组不存在，直接返回 0 即可
- 如果存在多个[test]组（例如：手工修改使存在多个[test]），则在位置靠前的组中增加本项并返回 1 即可
- 如果[test]组中的"起始值"已存在,则不能重复增加,直接返回 0 即可(如果第一个[test]有"起始值"存在，后面还有[test]组，但是无"起始值"存在，也认为已存在)
- 如果组名为 NULL（例：fcfg.item_add(NULL, "起始值"),则：
  - 如果是含组名的配置文件（任一行去除头尾空格后是“[***]”），直接返回 0 即可
  - 如果是简单配置文件，则检查整个文件中该项是否存在，若不存在，则加在简单配置文件的最后一行（注：含组名的配置文件是该组第一项）并返回 1，若存在，直接返回 0 即可
- 不考虑其它数据类型

★ int item_del(const char *group_name, const char *item_name)

使用说明：

在测试用例中调用 fcfg.item_del("test", "起始值")，则表示在配置文件的[test]组中删除"起始值=***" 项

- 删除成功返回 n(n>0)，表示删除了 n 个"起始值"项，否则返回 0
- 如果[test]组不存在，直接返回 0 即可
- 如果[test]组存在，但要删除的"起始值"项不存在，则直接返回 0 即可
- 如果[test]组存在，但要删除的"起始值"项重复存在（例如：手工修改使存在多个"起始值"），则删除该组所有同名项并返回 n（注意：不能删除其它组的同名项）
- 如果有多个[test]组，则只删除第一组中的"起始值"项，不存在则直接返回 0，不考虑后面的[test]组
- 删除 item 时，所在行全部删除，包括前面可能存在的空格/tab 和后面的注释等
- 如果组名为 NULL（例：fcfg.item_del(NULL, "起始值");则表示在简单配置文件中删除所有同名项（对于含组名的配置文件，则忽略组名，即删除该文件中所有组中的"起始值=***" 项并返回删除项数 n）

★ int item_update(const char *group_name, const char *item_name, const int item_value)

（共 6 个 item_update 重载函数，此处略，具体请看 class_ctf.h）

使用说明：

1、假设在测试用例中有：

```
int i = 12345;
fcfg.item_update("test", "起始值", i);
```

则表示在配置文件的[test]组将"起始值=***"项更新为"起始值=12345"；

若"起始值"项不存在，则在该组的指定位置处加入"起始值=12345"项

2、假设在测试用例中有：

```
double d = 123.45;
fcfg.item_update("test", "起始值", d);
```

则表示在配置文件的[test]组将"起始值=***"项更新为"起始值=123.45"；

若"起始值"项不存在，则在该组的指定位置处加入"起始值=123.45"项

3、假设在测试用例中有：

```
char *s="好日子";
fcfg.item_update("test", "起始值", s);
```

则表示在配置文件的[test]组将"起始值=***"项更新为"起始值=好日子"；

若"起始值"项不存在，则在该组的指定位置处加入"起始值=好日子"项

4、假设在测试用例中有：

```
string s="好日子";
fcfg.item_update("test", "起始值", s);
```

则表示在配置文件的[test]组将"起始值=***"项更新为"起始值=好日子"；

若"起始值"项不存在，则在该组的指定位置处加入"起始值=好日子"项

5、假设在测试用例中有：

```
char c = 'Y';
fcfg.item_update("test", "起始值", c);
```

则表示在配置文件的[test]组将"起始值=***"项更新为"起始值=Y"；

若"起始值"项不存在，则在该组的指定位置处加入"起始值=Y"项

6、假设在测试用例中有：

```
fcfg.item_update("test", "起始值");
```

则表示在配置文件的[test]组将"起始值=***"项更新为"起始值="；

若"起始值"项不存在，则在该组的指定位置处加入"起始值="项

- **注：“指定位置”**的定义在 enum class InsertPosition 中，共三种形式

- 更新/新增成功返回 1，否则返回 0
- 如果[test]组不存在，直接返回 0 即可
- 如果存在多个[test]组（例如：手工修改使存在多个[test]），则在位置最靠前的组中更新/增加本项并返回 1 即可（后续的同名组不处理）
- 如果[test]组存在，但要更新的项"起始值"重复存在（例如：手工修改使存在多个"起始值"），则更新位置靠前的一项并返回 1 即可（本组的后续其它同名项不处理）
- 更新项前后的数据类型允许不同（例："起始值=Y"更新为"起始值=12345",反之亦可）
- 更新时，整行替换，包括前面可能存在的空格/tab 和后面的注释等

- 如果组名为 NULL（例：fcfg.item_update(NULL, "起始值"),则：
  - 如果是含组名的配置文件（任一行去除头尾空格后是“[***]”），直接返回 0 即可
  - 如果是简单配置文件，则检查整个文件中该项是否存在，若不存在，则加在简单配置文件的最后一行（注：含组名的配置文件是该组第一项）并返回 1，若存在，替换该行并返回 1 即可（如有多项，则更新位置最靠前的第一项，后续同名项不处理）
- 不考虑其它数据类型

★ int item_get_value(const char *group_name, const char *item_name, int &item_value)
（共 6 个 item_get_value 重载函数，此处略，具体请看 class_cft.h）
使用说明：
1、假设在测试用例中有：
```
int i;
fcfg.item_get_value("test", "起始值", i);
```
如果配置文件的[test]组有"起始值=12345"，则调用后 i 值是 12345
2、假设在测试用例中有：
```
double d;
fcfg.item_get_value("test", "起始值", d);
```
如果配置文件的[test]组有"起始值=123.45"，则调用后 d 值是 123.45
3、假设在测试用例中有：
```
char s[80];
fcfg.item_get_value("test", "起始值", s);
```
若配置文件的[test]组有"起始值=今天是个好日子"，则调用后 s 值是"今天是个好日子"
4、假设在测试用例中有：
```
string s;
fcfg.item_get_value("test", "起始值", s);
```
若配置文件的[test]组有"起始值=今天是个好日子"，则调用后 s 值是"今天是个好日子"
5、假设在测试用例中有：
```
char c;
fcfg.item_get_value("test", "起始值", c);
```
如果配置文件的[test]组有"起始值=Y"，则调用后 c 的值是'Y'
6、假设在测试用例中有：
```
fcfg.item_get_value("test", "起始值");
```
如果配置文件的[test]组有"起始值=***"（任意项），则调用后函数返回 1，否则返回 0
- 取值成功返回 1，否则返回 0**（返回 0 时，前五个函数的第三个参数的值不可信）**
- 如果[test]组不存在，直接返回 0 即可，不要改变传入的 item_value 的值
- 如果存在多个[test]组（例如：手工修改使存在多个[test]），则在位置最靠前的组中取本项的值，根据存在与否返回相应值，不再考虑后续同名组
- 如果[test]组中"起始值"不存在，直接返回 0 即可，不要改变传入的 item_value 的值
- 如果[test]组中"起始值"存在多项，则取位置最靠前的项即可，不再考虑后续项
- 当使用两参的 item_get_value 时，不做任何操作，返回 1/0 即可

- 如果文件中存在能和 item 匹配但无=的情况（例："起始值 abc"），直接返回 0 即可
- 对于含组名的配置文件，如果组名为 NULL（例：fcfg.item_get_value(NULL,"起始值")，则直接返回 0；对不含组名的配置文件，则取位置最靠前的项即可
- 因为数据类型错误导致运行出错，不算错误（例如：用 short 变量取 int 值/float 变量取 double 值/int 变量取 double 值，给的一维字符数组的长度不足以容纳整个字符串等）

综合题 5-2：从 cft 派生出新类 cft_gai（gai= get_all_utems），适应另一种形式的配置文件

```
# 上学期输入输出与管道运算中用到的某些样例

[3-b4-01]
9999999999.99

[3-b4-02]
9999999999.90

[5-b18-ok-01]
请输入密码长度(12-16)， 大写字母个数(≥2)， 小写字母个数(≥2)， 数字个数(≥2)， 其它符号个数(≥2)
12 2 2 2 2
Ynk-Fq89oE-I
6q2ZApV$bD*@
Gy$@wnQ4BbP6
PXy%6*e9#o^2
9i@PH%pAlmnk
Su1Tr4!CT*kS
mx*-6p+Sl2gT
j@1r,2VCb?DI
?oAun7V%!97_
31yR,&D4.opV
```

%是有效字符

特点：
1、仍然用[组名]形式区分不同组
2、各项不是 name = value 的形式，每行都是有效数据
3、有效数据中可能出现#;（注释符）为有效字符的情况
4、仍然要排除每行首尾的空格/tab，以及仅含空格 tab 的空行（可能不适用某些特殊情况）

【工具函数集的定义(C++方式)】

class cft_gai 的定义放在 class_cft_gai.h 中，各成员函数的说明如下（假设 cft_gai fcfg）：

★ 从 class cft 中继承，可以使用的相关函数
- open / is_open / close
- set_comment_switch /get_comment_switch
- set_insert_position /get_insert_position
- group_add / group_del

★ int item_add(const char *group_name, const char *item_value)
（共 2 个 item_add 重载函数，此处略，具体请看 class_cft_gai.h）
使用说明：将 item_value（纯字符串形式，包括多行）写入 group_name 组的指定位置处

★ int item_del(const char *group_name, const char *item_value)
（共 2 个 item_del 重载函数，此处略，具体请看 class_cft_gai.h）
使用说明：将 group_name 组中的 item_value（纯字符串形式，整行匹配）所在行删除

★ int item_get_all(const char *group_name)
使用说明：将 group_name 组中的所有内容取出，每行一个 string（需释放空间）

【BigHW 新增目录要求：】

1、BigHW 新增目录要求

- 在 BigHW 中新建项目 test_config_file_tools（注意：下划线）
- 在 BigHW 中新建项目 test_config_file_tools_get_all_items（注意：下划线）
- 给出 BigHW_3.rar 供参考(不要在 BigHW_3 中实现,而是将所需文件对应平移到 BigHW 中)，需要平移的文件列表如下：
  - class_cft.h
  - class_cft_gai.h
  - class_cft.cpp
  - class_cft_gai.cpp
  - lib_tcfg_tools.h
  - lib_tcfg_tools_get_all_items.h
  - lib_tcfg_tools.lib
  - test_config_tools.cpp
  - test_config_tools_get_all_items.cpp

2、完成 class_cft 的定义与实现

- 补充完整 class_cft.h
- 完成 class_cft.cpp 的实现
- 通过 test_config_tools.cpp 的测试

3、完成 class_cft_gai 的定义与实现

- 补充完整 class_cft_gai.h
- 完成 class_cft_gai.cpp 的实现
- 通过 test_config_tools_get_all_items.cpp 的测试

【测试用例的编写：】

1、每位同学需要编写一个测试用例（覆盖给出的 test_config_tools.cpp 即可），涉及的配置文件类型包括普通配置文件及简单配置文件；涉及的操作有新建、增加、删除、更新、读取；涉及的数据包括支持的所有类型

2、每位同学的测试用例生成的简单配置文件的文件名约定为“u1234567_s.cfg”，含组名的配置文件的文件名约定为“u1234567_g.cfg”（1234567 为学号，各人**对应替换**即可）

3、测试用例要求能够**同学间双向验证**（即你的 test_config_tools.cpp 和别人的 class cft 工具函数集的实现放入同一个项目中,运行结果与和你自己的 class cft 工具函数集的运行结果应一致，每人的公共函数集需要验证至少 3 人的 test_config_tools.cpp，将验证名单在 test-cft.cpp 源程序的第 2 行用注释说明即可（形式同前，如果查验不正确则要连环扣分）

【注意：】test_config_tools.cpp 源程序可以提供给别人，工具函数集不可以

4、附件 BigHW_3 中 test_config_tools.cpp 给出了一个不是很完善的测试用例供参考

5、test_config_tools_get_all_items.cpp 要求同上

【特别提示：】

1、读写方式打开文件时，如果在原来基础上变大，可以不调用改变文件大小的函数；如果在原来基础上缩小，则必须调用改变文件大小的函数

2、如果文件被重新写入（组增加/组删除/项增加/项减少）后，用 UltraEdit 打开时**自动切换**为 16 进制方式，则是因为文件中包含了非文本字符（例如：文件尾部多一串 0x00 等），**这种情况必须解决**，即更新后的配置文件必须是文本文件格式，不能包含非文本字符

3、文本文件有 Windows/Linux 两种格式，基本要求为支持 Windows 格式（能支持 Linux 最好）

【实现要求：】

1、不允许使用任何形式的全局变量/数组/指针，允许使用全局的宏定义或常变量
2、不允许使用 goto
3、不允许在函数中关闭文件并再次打开（已给出的 file_resize 例外）
4、不允许删除文件，包括临时文件
5、不允许拷贝文件，包括临时文件
6、**不限制 string 的使用**，但 STL 容器的相关结构仍然**禁用**

【提交要求：】

1、提交作业前，先做好完整备份
2、移除之前所有的lib，保证之前所有的项目都编译通过
3、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、**6 月 5 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

基于本次作业两个工具集的应用（可以合理预判，提前做  ）