【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 VS2019 编译通过即可（“0 errors, 0 warnings”）

3、 不允许使用 scanf/printf 进行输入/输出

综合题 5-1：完成一套配置文件读取的工具集

【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
;2021.05.24 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
;2021.05.24 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 的定义放在 cfg_file_tools.h 中，对各成员函数的说明如下（假设 CTF 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，这种打开方式只适用于 item_get_value 函数，若文件不存在，直接返回 NULL 即可
  - ◆ 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，请使用其它方式来标记打开配置文件正确/错误

★ 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

★ 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 cfg_file_tools.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]组的最开始处加入 "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]组的最开始处加入 "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]组的最开始处加入 "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]组的最开始处加入 "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]组的最开始处加入 "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]组的最开始处加入 "起始值=" 项（空项）
- 注：“最开始”-指本组[组名]的下一行
- 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(fp, 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 cfg_file_tools.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，则在该组的最开始处加入"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，则在该组的最开始处加入"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，则在该组的最开始处加入"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，则在该组的最开始处加入"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，则在该组的最开始处加入"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，则在该组的最开始处加入"起始值="项

- **注：“最开始”**-指本组[组名]的下一行
- 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(fp, 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 cfg_file_tools.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(fp, "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）

【BigHW 新增目录要求：】

- 在 BigHW 中新建一个项目 test-cft，要求最后形成的可执行文件名是 test-cft.exe(注：不做特殊设置的话，缺省就是此可执行文件名)

【附件给出的文件及存放目录：】

- common 目录
  - cfg_file_tools.cpp ：本工具集对应的源程序文件，目前为空，按需完成，每个函数的具体要求见 cpp，**函数名及参数表不准改动**
- include 目录：
  - cfg_file_tools.h ：本工具集对应的头文件，已给出部分，按限制要求补充完成
- test-cft 目录：
  - test-cft.cpp：本工具集的测试用例（不完整，留待补充），作业完成过程中可根据自己的需要随意修改测试内容
- test-cft.exe ：给出的测试用例运行的可执行文件供参考

【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-cft.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-cft.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 tes-cft.cpp，将验证名单在 test-cft.cpp 源程序的第 2 行用注释说明即可（如果查验不正确则要连环扣分）

【注意：】test-cft.cpp 源程序可以提供给别人，工具函数集不可以

4、附件的 test-cft.cpp 给出了一个不是很完善的测试用例供参考

【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

【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、**不再限制 string 的使用**

7、**要能支持 Linux 格式的配置文件**（可将现有配置文件通过 17-b2 的 wtol 转换后进行测试）

综合题 5-2：在已经完成 90-02-b4 的基础上，完成基于配置文件的增强版 LED 显示屏模拟程序

【基本作业要求 - 在已经完成 90-02-b4 的基础上，再做如下要求：】

1、只要求 hdc 工具方式，不再要求小字号的*显示形式

2、要考虑有多条内容需要显示的情况

3、要考虑一条内容大于屏设置大小的情况（例：屏幕 4 行 8 字，显示的某条内容为 100 个汉字）

4、显示内容中允许出现半角的英文字母及标点符号，半角的英文字母及标点符号在汉字库中没有对应关系，在没有英文字库的情况下，要求将半角转为对应全角字符后显示（例：内容中有半角 A，则需要显示全角 A ）

- 需要转换的半角字符一共 94 个，即 ASCII 码 0x21-0x7E 间所有字符（在键盘上均能找到）
- 可以定义转换表（例：A(ASCII 码 0x41) => A (区位码 0333)），到时直接查表即可

5、显示内容中允许出现非 GB2312 的汉字（GBK 或 GB18030，例如：喆），这些汉字在给出的汉字库是没有的，显示时忽略即可

- 如何判断哪些汉字属于 GB2312
- 对于非 GB2312 的多字节（2-4 字节）汉字，如何才能跳过所有字节

6、LED 屏显示的大小、颜色、内容等从同目录下的配置文件 led.cfg 中读取，配置组及配置项名称要求保持一致，不准改动，配置值可变，样例及说明如下：

```
[全局设置]
行数=5
列数=10
背景色=00FF00
特效 1=Y
特效 2=Y
特效 3=Y
特效 4=Y
屏延时=2
条延时=3
字库=HZK16F

[内容设置]
item1_color=0000FF
item1=***
#item2_color=FF00FF
item2_color=0000FF
item2=***
item3=***
```

- 程序支持默认值，若配置文件中未给出此项/此项的值不在合理范围内，则用默认值代替
- 行数的值为汉字的数量，范围[1..5]，默认值 4
  - **默认值的解释（下同）：**如果[全局设置]组中无“行数”项，则默认为 4；有“行数”项，但值不在[1..5]之间，则默认为 4
- 列数的值为汉字的数量，范围[6..12]，默认值 8
- 背景色的表示为至少 6 位 16 进制数，排列顺序为 RGB，每种颜色的取值范围为[00..FF](即十进制 0-255)。默认为 000000（纯黑色）
  - 超过 6 位则取低 6 位（例：背景色=00FF00FF，则等价于 FF00FF，即黄色）
  - 不足 6 位/每位不是[0..9/a..f/A..F]则认为错误，取默认值即可
  - **提示：**从配置文件中读取背景色时，应选用哪种数据类型？（要求 class CFT 中的重载函数 get_item_value 不增加新类型）

- 特效编号范围 1-20，顺序递增
  - 1-4 为上次作业的基础显示及三种特效方式（三种特效的编号顺序自行决定）
  - 如果还有新的的特效，从 5 开始递增即可（也可以不连续编号）
  - 值 Y/N 代表是否启用该种特效（例：“特效 2=N”，则后续显示中不要特效 2）
  - 特效编号在配置文件中的排列可乱序（例：“特效 3”在“特效 1”的前面）
  - 如果配置文件中该编号对应的特效未实现，改为缺省 1 即可（例：“特效 5=Y”，但实际未实现特效 5，则使用时直接用 1 即可）
  - 配置文件中特效编号重复的，以前面一个为准决定是否使用
- 屏延时指当某条内容大于屏设置大小时（例：屏幕 4 行 8 字，显示的某条内容为 100 个汉字），多屏内容切换时的延时，范围[0..10]，默认 2，单位秒
  - 某些特效可能两屏间不需要延时，例如：从右到左横幅拉动形式，忽略即可
- 条延时指两条内容间的切换延时（上条结束~下条开始之间的停顿），范围[0..10]，默认 3，单位秒
- 字库设置项决定读哪个字库，仅要求支持 GB2312 简体/繁体即可，缺省简体
- item 的编号范围 1-20，每个 item 最长 128 个汉字，超出则截断即可
  - item 编号可以不连续，配置文件中的排列可无序，重复则前一个为准
  - item 中的英文字母及符号要转为全角，非 GB2312 的汉字要忽略
  - 程序运行时，按 item 编号从 1-n 循环往复并依次显示，每个 item 可能分若干屏，如有编号空缺，跳过即可
  - 如果配置文件中一个 item 也没有，则显示“Welcome”
  - 每个 item 的显示特效采用在有效特效编号（置 Y）中随机选择的方式
  - **特别约定：**半角的;和#，是配置文件的注释项，因此不允许出现在正文中，如果出现，优先解释为注释项
- item_color 的编号与 item 对应
  - 颜色表示方法同[全局设置]组的“背景色”项，但默认值为 FFFFFF（纯白色）
  - 如果某个 item 对应的 item_color 不存在，则取默认值
  - 如果认为设置颜色与背景色一致导致显示异常，不是你的错

【实现要求：】

1、在 BigHW 中新建一个项目 90-02-b4-pro**（千万不要错!!!，02 代表第二学期，b4-pro 代表第五个大作业，pro 是纯小写）**，要求最后形成的可执行文件名是 90-02-b4-pro.exe(注：不做特殊设置的话，缺省就是此可执行文件名)

2、遵循之前大作业模板使用说明中的要求,即本项目单独用的源程序文件名放在 90-02-b4-pro 中，需要共享的放在 common/include 中，需要的库文件放在 lib 中

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 90-01-b*/90-02-b*能编译通过并运行正确

5、整个程序，**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

6、整个程序，**不允许**使用 goto，**不限制** string 类的使用

7、hdc 工具集仍然可以做自己的和老师的相互切换，方法不变（同 90-02-b3）

【提交要求：】

1、提交作业前，先做好完整备份

2、将lib_thdc_tools.lib文件先从90-02-b3/90-02-b4项目中移除，再从lib目录中删除，在此情况下要保证编译能通过（即编译形成的exe文件一定要用自己完成的工具函数集）

3、**删除两个汉字库文件（之前的90-02-b4目录下的也对应删除）**

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

【实验报告：】

本次作业暂时不需要提交单独的实验报告

【编译器要求：】

仅 VS2019 通过即可

【作业要求：】

1、6 月 20 日（第 16 周周日）前网上提交本次作业（本次作业时间为两周）

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

【提示：】

1、 不要卡 DDL!!!

2、 本截止日期为本课程作业的最终提交日期，之后作业提交系统会关闭，考虑到作业的批改需要预留一定的时间，不接受任何方式的延期请求（包括有正式病假条及合理事假理由在内的任何理由）