CS111，C Programming

Lab / Others

黄嘉炜

huangjw3@mail.sustech.edu.cn

Outline

- Review
- Others – enum
- Others - include, header (.h) & source (.c) file
- Assignment

Review: Extract keywords from file

Write a program to read text file (ASCII encoding), extract words insides which **case insensitive**, and seperated by space, Enter. And target words should **NOT** be given stop-words. Finally, the extracted words need to be sorted by word occurrence. And print out Top-K words in descending order.

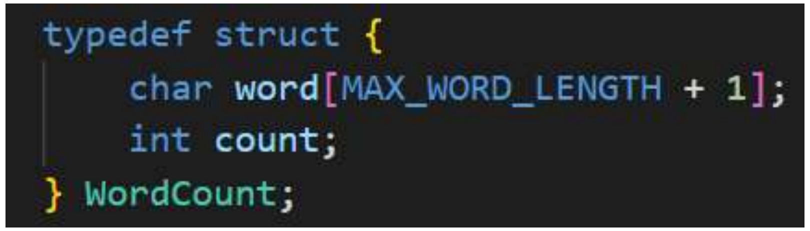
Input

- 1st line: the path of input file. The length of path <= 100. The count of distinct words in input file, less than 10,000. And maximal length of each word is less than 50.
- 2nd line: the path of stop-words file. Each line contain 1 stop-word. The length of path <= 100. The count of stop-words in 1st line of stop-words file. The content format of stop-words file is same as lab example.
- 3th line: K, the number of word for output (aka, Top-K).

Output

- Each output line contains: word, and it's occurrence. This 2 fields are sperated by 1 space.
- Words should be **lowercase**, and also exclude following 4 special characters: `'('`, `')'`, `','`, `'.'`
- Note that: Only output K words, even throught (K+1)th word is same occurrence with the Kth word, and then consider of sub-order by alphabet in ascending (which `strcmp` can help).

Review: Extract keywords from file



```
typedef struct {
    char word[MAX_WORD_LENGTH + 1];
    int count;
} WordCount;
```

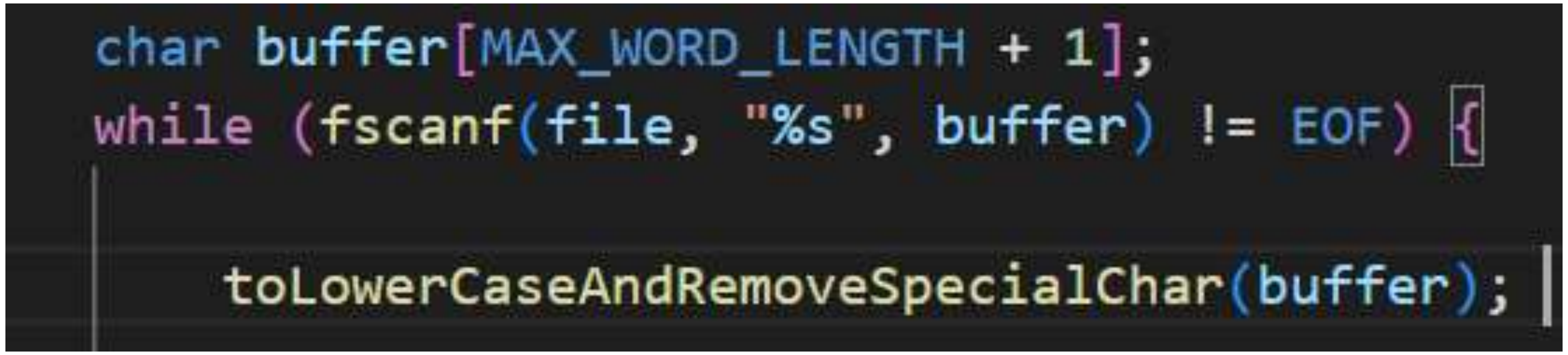
解题思路

- Step1，读取文件 stopwords.txt，构建 char stop_words[200][51], 记录停用词
- Step2，打开文件 lab11_test{X}.txt (目标分析文件)，得到 File* file
- Step3，定义词语列表：WordCount word_array[10000]，记录目标文件的词语和出现次数
- Step4，从 file 中读取一个词语 char word[51]
- Step5，将 read_word 中字符全转为小写，并剔除特殊字符，检查是否在 stop_words 中
  - 如果 read_word 不在 stop_words 中、并且不在 word_array 中，添加到 word_array，出现次数记为1
  - 如果 read_word 不在 stop_words 中、并且在 word_array，出现次数+1
- Step6，重复 Step4 ~ 5，直到读到文件结尾（EOF）
- Step7，对 word_array 进行二维排序（出现次数倒序 + 字母顺序）
- Step8，遍历打印输出 word_array[{top_k}]

Review: Extract keywords from file

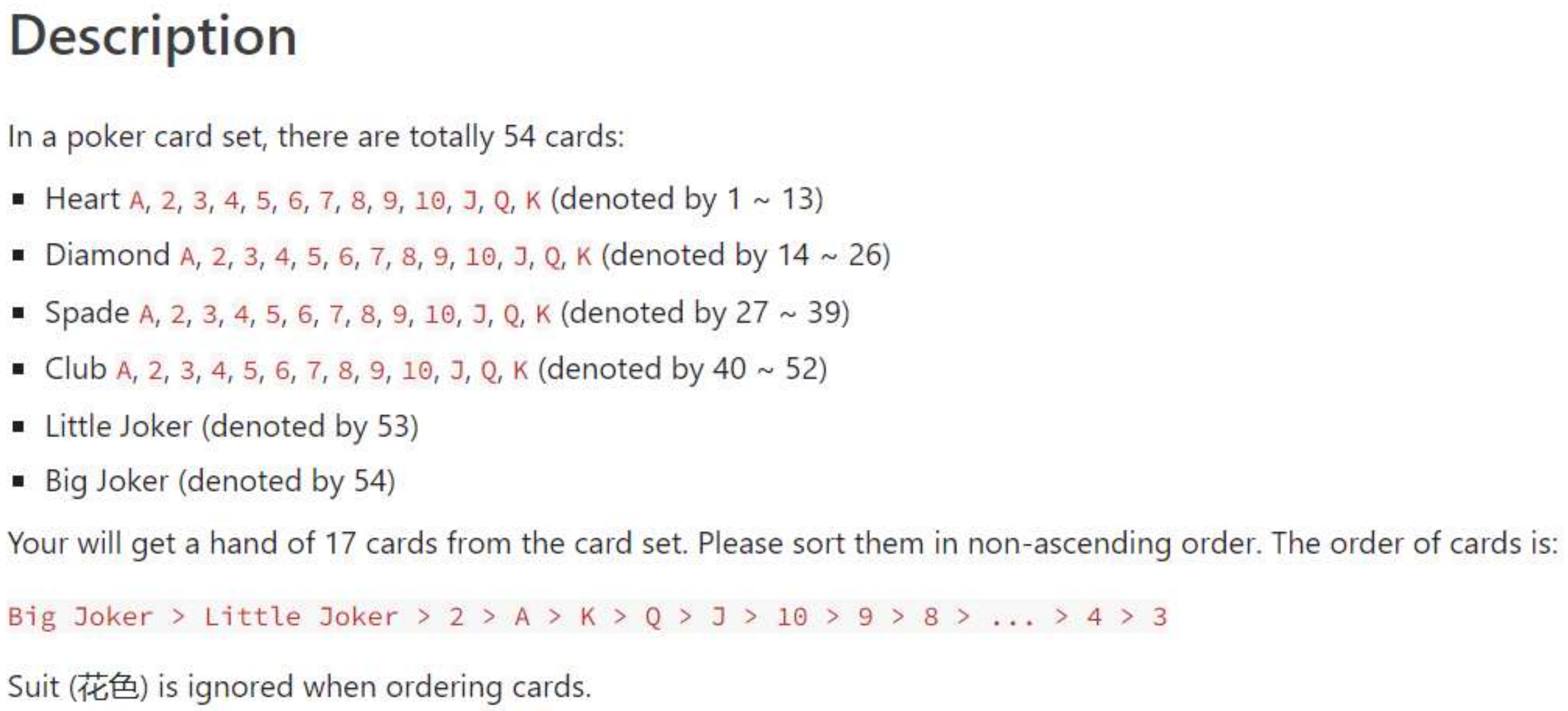
Step4 ~ 5 出现的问题

- Step4，从 file 中读取一个词语 char word[51]
- Step5，将 read_word 中字符全转为小写，并剔除特殊字符，....



```c
char buffer[MAX_WORD_LENGTH + 1];
while (fscanf(file, "%s", buffer) != EOF) {

    toLowerCaseAndRemoveSpecialChar(buffer);
```

Review: Poker Card



Description

In a poker card set, there are totally 54 cards:

- Heart `A`, `2`, `3`, `4`, `5`, `6`, `7`, `8`, `9`, `10`, `J`, `Q`, `K` (denoted by 1 ~ 13)
- Diamond `A`, `2`, `3`, `4`, `5`, `6`, `7`, `8`, `9`, `10`, `J`, `Q`, `K` (denoted by 14 ~ 26)
- Spade `A`, `2`, `3`, `4`, `5`, `6`, `7`, `8`, `9`, `10`, `J`, `Q`, `K` (denoted by 27 ~ 39)
- Club `A`, `2`, `3`, `4`, `5`, `6`, `7`, `8`, `9`, `10`, `J`, `Q`, `K` (denoted by 40 ~ 52)
- Little Joker (denoted by 53)
- Big Joker (denoted by 54)

Your will get a hand of 17 cards from the card set. Please sort them in non-ascending order. The order of cards is:

```
Big Joker > Little Joker > 2 > A > K > Q > J > 10 > 9 > 8 > ... > 4 > 3
```

Suit (花色) is ignored when ordering cards.

Review: Poker Card

Input

The input includes a single line consisting of 17 unique integers in the range $[1, 54]$, representing the 17 cards in your hand.

Output

Print the sorted cards in a single line

- For Big Joker, print `G`
- For Little Joker, print `g`
- For other cards, ignore their suit and print the index only.

For the same cards (ignoring suits), put no space between them. That is to say, we want `QQ JJJ 101010 9 888` instead of `Q Q J J J 10 10 10 9 8 8 8`.

Review：Poker Card

解题思路：桶排序

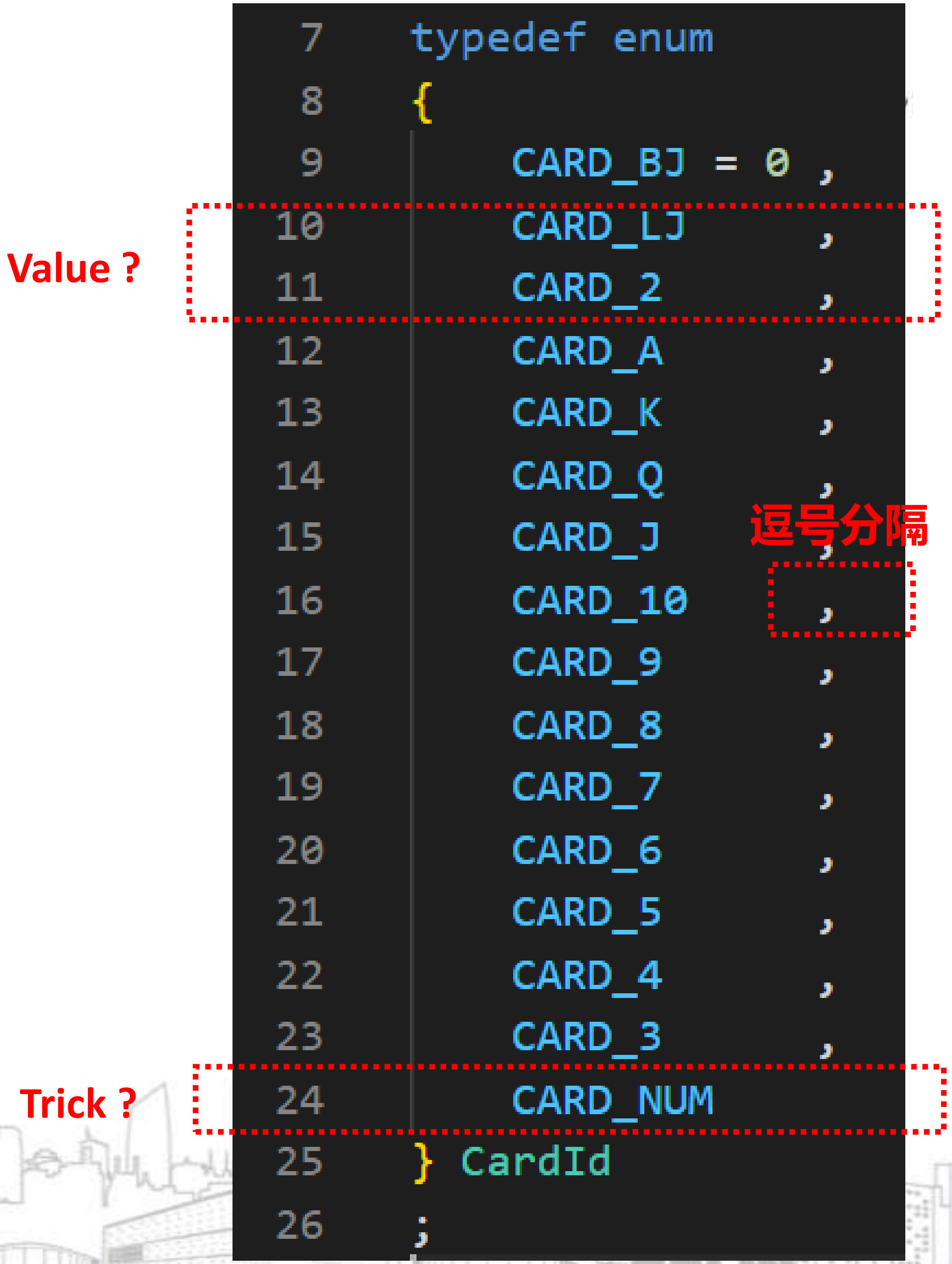
- Step1, 构建 int card_array[15], 分别记录每张牌出现的次数（初始化均为0）
  - Position 0th → big joke; 1st -> little joke; 2nd -> “2”; ….
  - 可将 position 理解为 card_id
- Step2, 读取输入 card_value (数值范围 1 ~ 54) ,
- Step3, 将 card_value， 映射成 card_id （即 card_array 位置）， 进而更新 card_array
- Step4， 重复 step2 ~ 3, 直到读取到 17 张牌
- Step5， 遍历打印输出 card_array （注意，card_name 与 position / card_id 一一对应）

About: enum

Enum is a **user defined data type** in C,

assign names to integer constants,

for a program **easy to read and maintain**.



```
typedef enum
{
        CARD_BJ = 0 ,
        CARD_LJ     ,
        CARD_2      ,
        CARD_A      ,
        CARD_K      ,
        CARD_Q      ,
        CARD_J      ,
        CARD_10     ,
        CARD_9      ,
        CARD_8      ,
        CARD_7      ,
        CARD_6      ,
        CARD_5      ,
        CARD_4      ,
        CARD_3      ,
        CARD_NUM
} CardId
;
```

About: include, header (.h) & source (.c) file

A header file is a file with extension **.h (e.g, stdio.h)**,

which contains function declarations and macro /

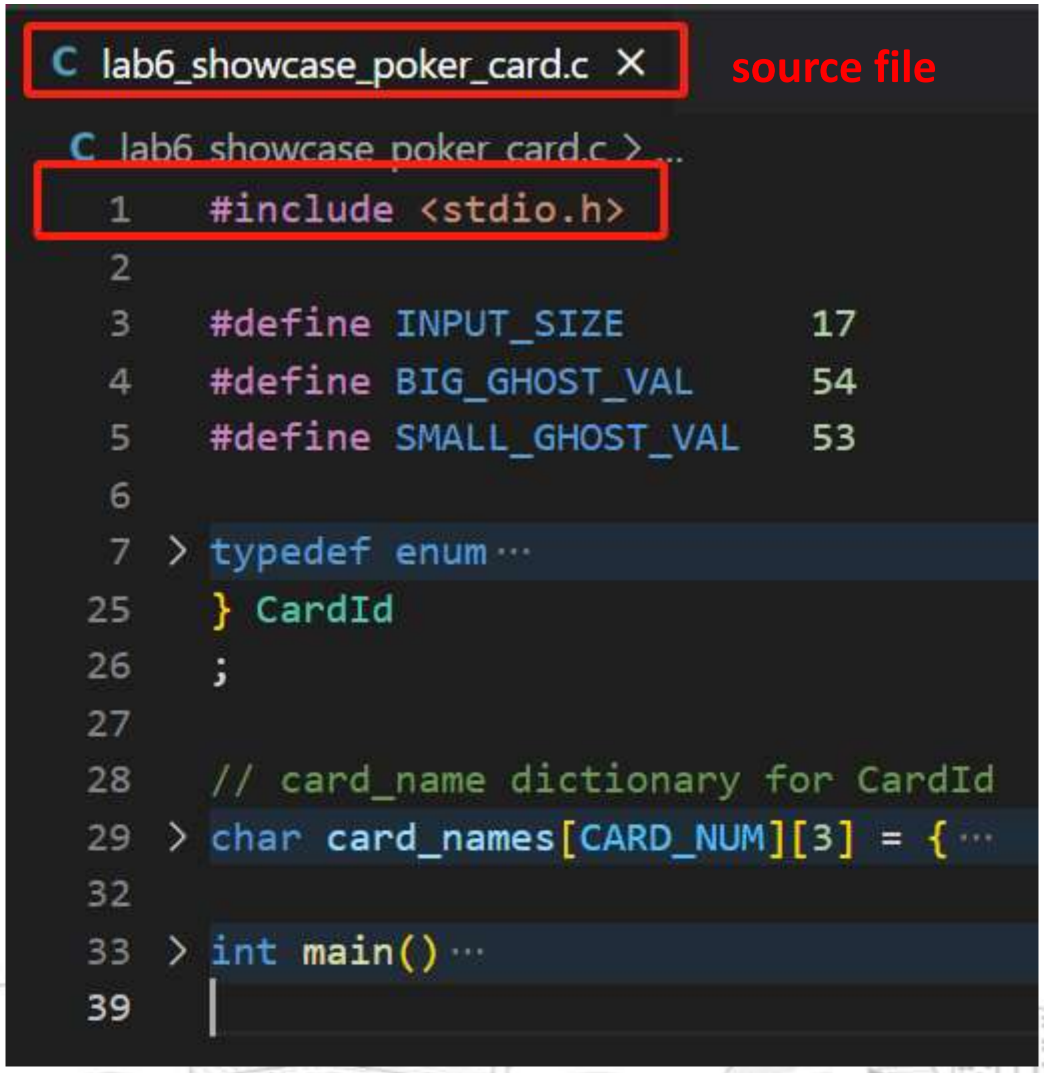
data-type definitions,

that can be used by different source files.

2 types of head file

- C compiler defined, e.g. stdio.h,
  - #include <xxx.h>
- User defined …
  - #include “xxx.h”

why ?
When project is getting complicated …



lab6_showcase_poker_card.c ×

source file

lab6_showcase_poker_card.c > ...

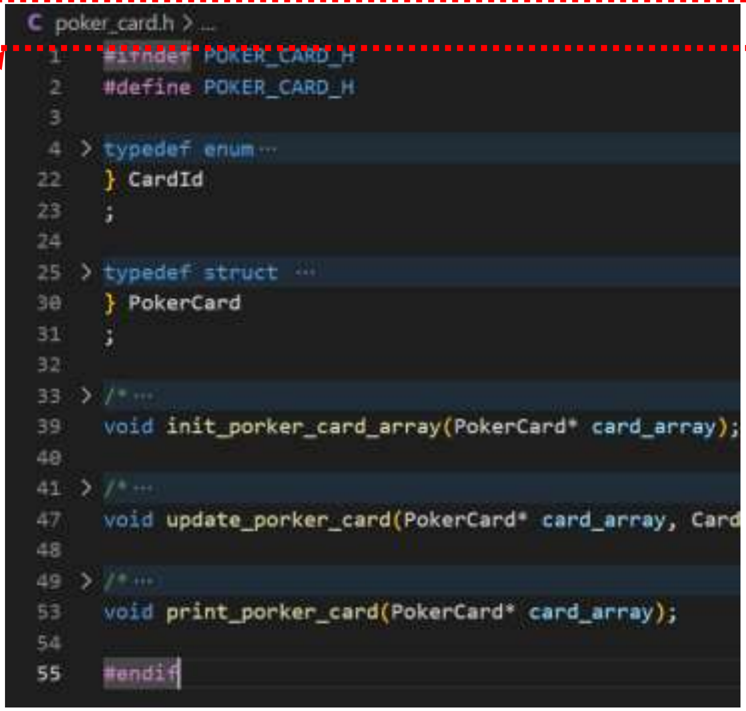
```c
#include <stdio.h>

#define INPUT_SIZE          17
#define BIG_GHOST_VAL       54
#define SMALL_GHOST_VAL     53

> typedef enum ···
} CardId
;

// card_name dictionary for CardId
> char card_names[CARD_NUM][3] = { ···

> int main() ···
```

About: include, header (.h) & source (.c) file

When project is getting complicated …

⇒ 1 single c file will break down …

⇒ Note that: still only 1 entry point (main(…))

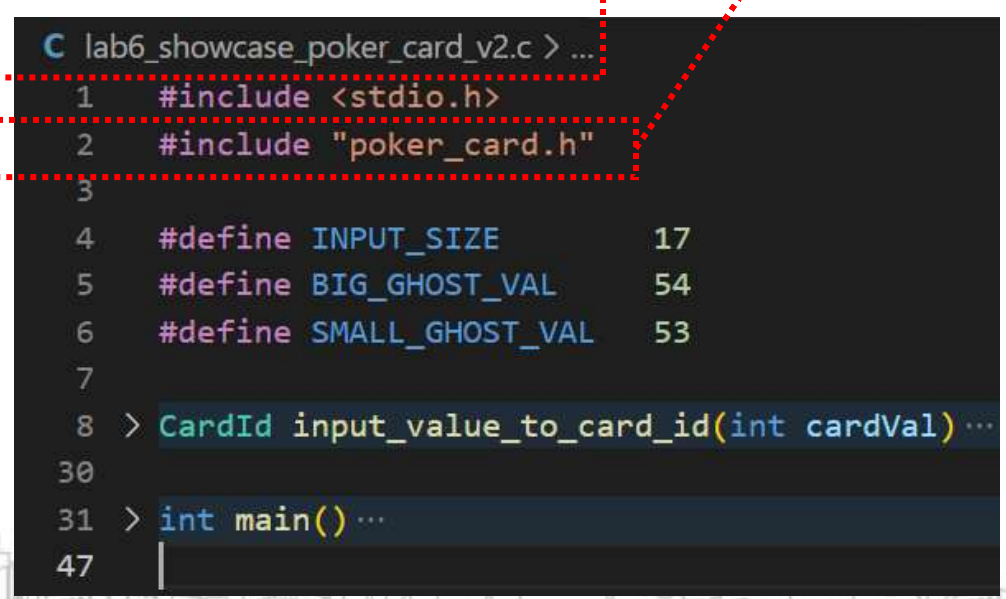


poker_card.h > ...

```c
#ifndef POKER_CARD_H
#define POKER_CARD_H

typedef enum ···
} CardId
;

typedef struct ···
} PokerCard
;

/* ···
void init_porker_card_array(PokerCard* card_array);

/* ···
void update_porker_card(PokerCard* card_array, Card

/* ···
void print_porker_card(PokerCard* card_array);

#endif
```

User defined head file:
poker_card.h
Which can be included by:
lab6_showcase_poker_card_v2.c

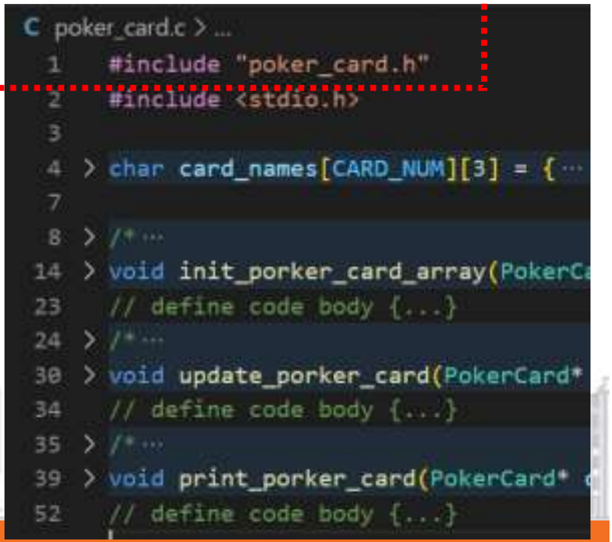
The functions declared here can be called by:
lab6_showcase_poker_card_v2.c



lab6_showcase_poker_card_v2.c > ...

```c
#include <stdio.h>
#include "poker_card.h"

#define INPUT_SIZE        17
#define BIG_GHOST_VAL     54
#define SMALL_GHOST_VAL   53

CardId input_value_to_card_id(int cardVal) ···

int main() ···

```

Another source file:
poker_card.c
To define the function declarations in
poker_card.h



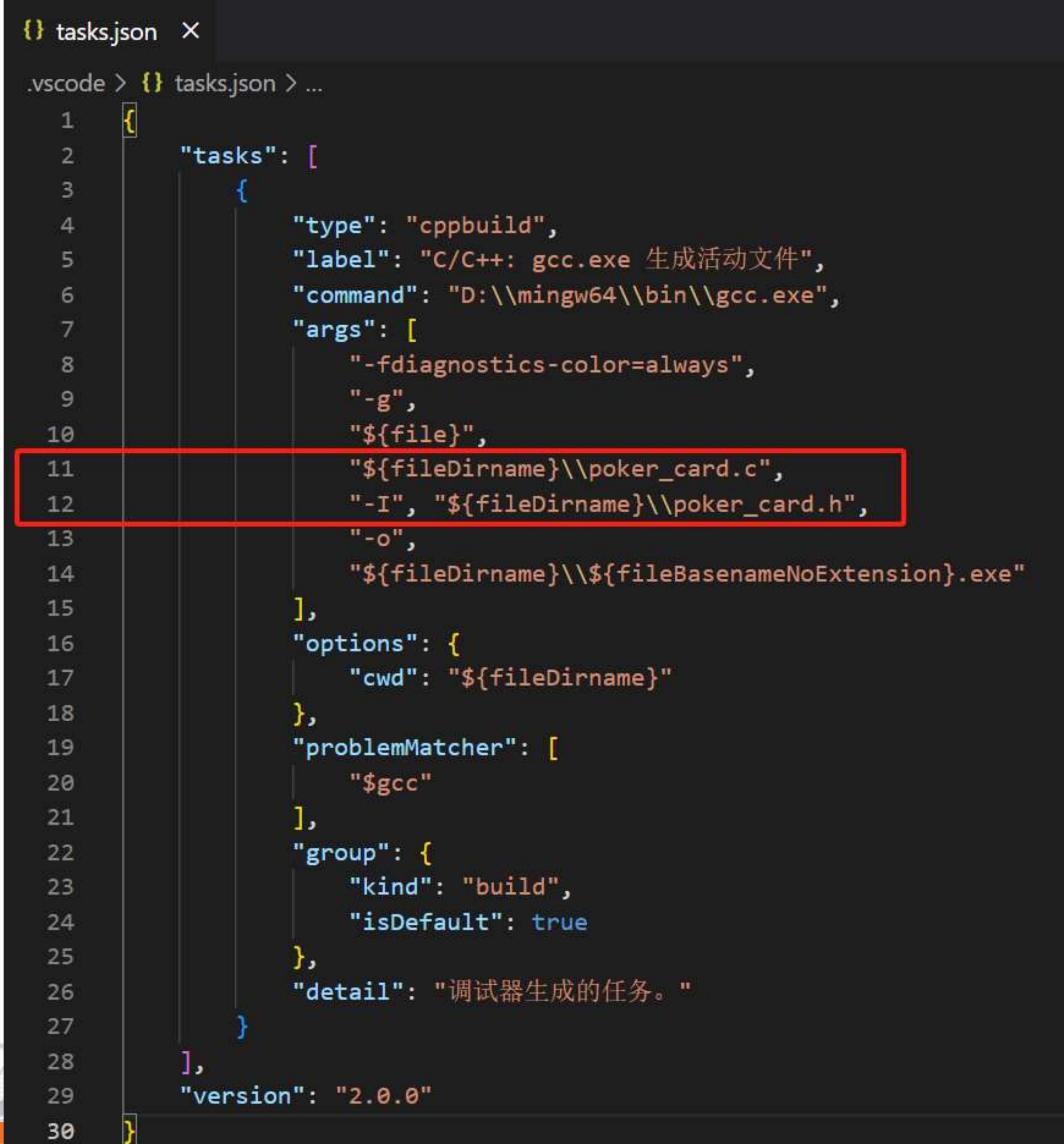
poker_card.c > ...

```c
#include "poker_card.h"
#include <stdio.h>

char card_names[CARD_NUM][3] = { ···

/* ···
void init_porker_card_array(PokerCa
// define code body {...}
/* ···
void update_porker_card(PokerCard*
// define code body {...}
/* ···
void print_porker_card(PokerCard* c
// define code body {...}
```

Appendix: VS Code

debug with multi-files ?

将自定义的 头(.h)文件 和 source(.c) 文件 (如：poker_card.h, poker_card.c) 放到代码目录下，并将其添加到编译任务中（tasks.json）



```json
{
    "tasks": [
        {
            "type": "cppbuild",
            "label": "C/C++: gcc.exe 生成活动文件",
            "command": "D:\\mingw64\\bin\\gcc.exe",
            "args": [
                "-fdiagnostics-color=always",
                "-g",
                "${file}",
                "${fileDirname}\\poker_card.c",
                "-I", "${fileDirname}\\poker_card.h",
                "-o",
                "${fileDirname}\\${fileBasenameNoExtension}.exe"
            ],
            "options": {
                "cwd": "${fileDirname}"
            },
            "problemMatcher": [
                "$gcc"
            ],
            "group": {
                "kind": "build",
                "isDefault": true
            },
            "detail": "调试器生成的任务。"
        }
    ],
    "version": "2.0.0"
}
```

Assignment ） 知识点梳理 + 节选代码

回顾之前所学的C语言知识，

请从之前作业代码中选取与以下关键语法点相关的部分，

并给出相应的代码片段和说明。

需涵盖以下知识点：

- 变量 和 数据类型
- 条件语句 和 循环语句
- 数组
- 指针
- 函数
- 结构体

格式： PDF

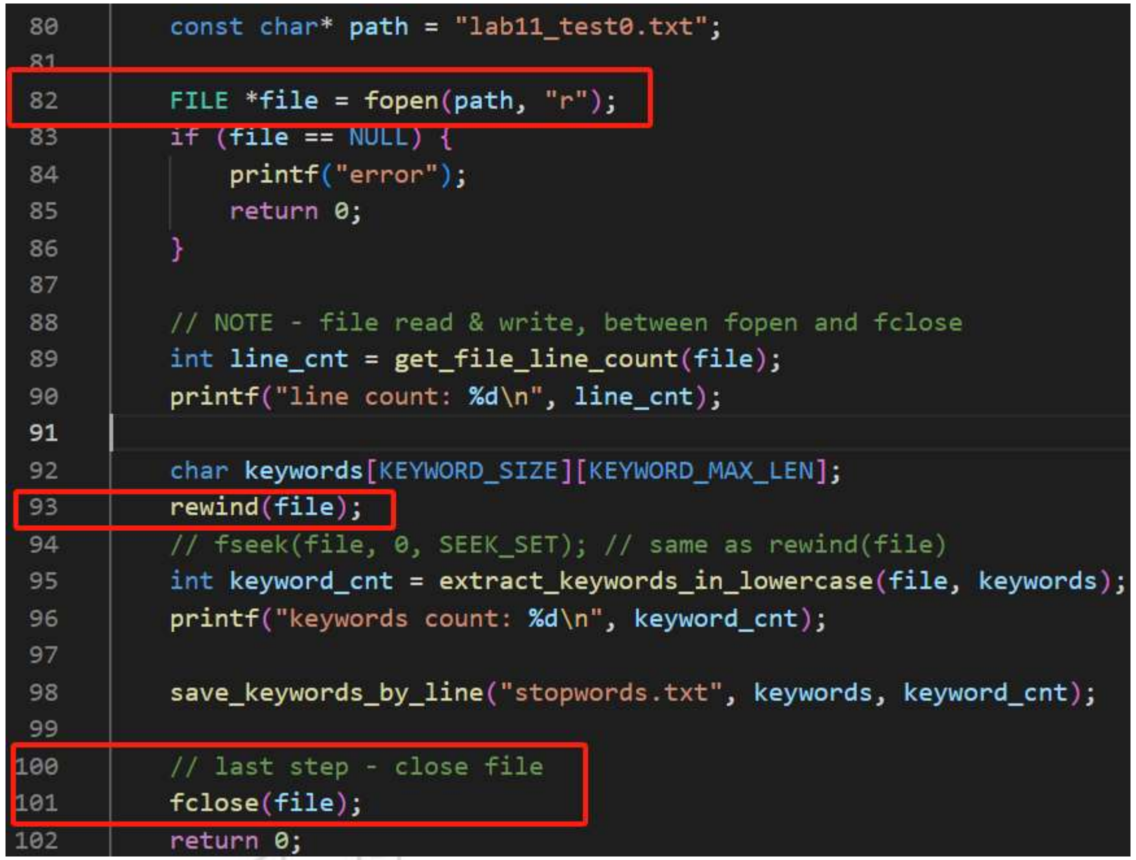
提交： Blackboard

Due： 2024/6/21

Assignment ） 以 File 为例

文件读写

- 首先，通过文件路径，进行文件打开 fopen(...), 得到 File* （File指针）
    - 注意：mode选择，“r” （读...） / “w” (写...)
- 通过 File*，进行文件读写操作
    - 读的方法：fgets(...), fscanf(...)
        - 重新读取需要执行 rewind(...) 或 fseek(...)
    - 写的方法：fputs(...), fprintf(...)
- 最后，需要 fclose(...), 进行文件关闭

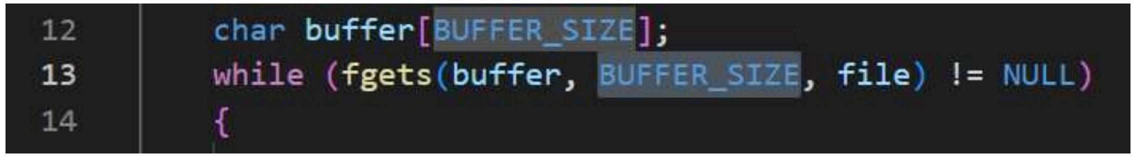


```c
    const char* path = "lab11_test0.txt";

    FILE *file = fopen(path, "r");
    if (file == NULL) {
        printf("error");
        return 0;
    }

    // NOTE - file read & write, between fopen and fclose
    int line_cnt = get_file_line_count(file);
    printf("line count: %d\n", line_cnt);

    char keywords[KEYWORD_SIZE][KEYWORD_MAX_LEN];
    rewind(file);
    // fseek(file, 0, SEEK_SET); // same as rewind(file)
    int keyword_cnt = extract_keywords_in_lowercase(file, keywords);
    printf("keywords count: %d\n", keyword_cnt);

    save_keywords_by_line("stopwords.txt", keywords, keyword_cnt);

    // last step - close file
    fclose(file);
    return 0;
```

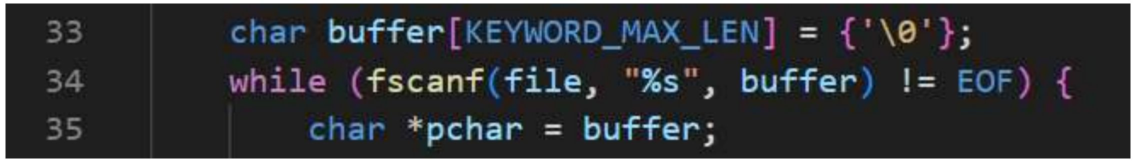
Assignment ） 以 File 为例

文件读写

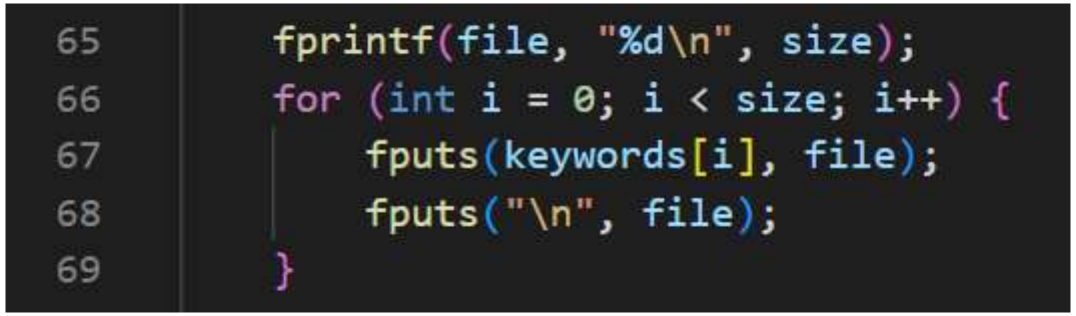
- 首先，通过文件路径，进行文件打开 fopen(...), 得到 File* （File指针）
    - 注意：mode选择，“r” （读...） / “w” (写...)
- 通过 File*，进行文件读写操作
    - 读的方法：fgets(...), fscanf(...)
        - 重新读取需要执行 rewind(...) 或 fseek(...)
    - 写的方法：fputs(...), fprintf(...)
- 最后，需要 fclose(...), 进行文件关闭



```
    char buffer[BUFFER_SIZE];
    while (fgets(buffer, BUFFER_SIZE, file) != NULL)
    {
```



```
    char buffer[KEYWORD_MAX_LEN] = {'\0'};
    while (fscanf(file, "%s", buffer) != EOF) {
        char *pchar = buffer;
```



```
    fprintf(file, "%d\n", size);
    for (int i = 0; i < size; i++) {
        fputs(keywords[i], file);
        fputs("\n", file);
    }
```

Southern University of Science and Technology

THANK YOU