§5.数组 - PART III

【注意:】

1、 本次作业不允许使用后续课程的知识点，包括但不限于指针、引用、结构体、类等概念!!!
2、 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，不允许使用 C++的 string 类
3、 cstdio 及 cmath 中的系统函数可以直接使用，包括课上未介绍过的，具体可自行查阅相关资料
4、 除明确要求外，所有 cpp 源程序不允许使用 scanf/printf 进行输入/输出
5、 多编译器下均要做到“0 errors, 0 warnings”
6、 部分题目要求 C 和 C++两种方式实现，具体见网页要求
7、 输出为浮点数且未指定格式的，均要求为 double 型，C++为 cout 缺省输出，C 为%lf 的缺省输出
8、 认真阅读格式要求及扣分说明!!!

【输出格式要求:】

1、 为方便机器自动判断正确性，作业有一定的输入输出格式要求（但不同于竞赛的无任何提示）
2、 每个题目见具体说明，必须按要求输入和输出，不允许有偏差
3、 没有特别说明的情况下，最后一行有效输出的最后有一个 endl

补充:

11、 用函数+数组方式重新完成 3-b5/4-b1(人民币转大写)，要求如下：

【要求:】1、所有的大写数字均放在全局一维只读字符数组 chnstr 中，具体形式为 const char chnstr[]="零壹贰叁肆伍陆柒捌玖";
凡需输出“零-玖”的地方，只允许从此数组中取值
2、其它大写内容（拾佰仟万亿圆角分整）可自行取值
3、转换后的内容不允许逐次输出，最后的输出只允许用一句输出语句来完成，具体分为两个小题
3.1 5-b11-1.c ：用全局 char result[256]; 存放转换结果，用 printf("%s\n", result);输出结果
3.2 5-b11-2.cpp ：用全局 string result; 存放转换结果，用 cout << result << endl;输出结果（本小题允许 string）
(注：输入提示、错误提示等个性化输出允许自行按需组织，但输出大写转换结果的语句只能是一句)

【提示:】1、根据分解的各位数字从 chnstr 中取部分内容
2、各位数字要输出的内容依次放入 result 中，最后输出这个字符串即可
3、保证 3-b5 中的所有测试数据均通过，输入输出格式要求同 3-b5（不考虑输入错误）
4、如果在 Dev C++中有某个警告消除不掉，允许将全局一维只读数组改为 const char chnstr[]="零壹贰叁肆伍陆柒捌玖拾"（未碰到则忽略此提示即可）

12、 用一维字符数组方式实现下列函数

函数原型	功能说明	返回值
int tj_strlen(const char str[]);	求字符串 str 的长度	字符串长度
int tj_strcat(char s1[], const char s2[]);	将字符串 s2 追加到 s1 后面，含\0	0
int tj_strncat(char s1[], const char s2[], const int len);	将字符串 s2 的前 len 个字符追加到 s1 后面，并添加\0 ★ 若 len 比 s2 的长度大，则追加整个 s2 即可(含\0)	0
int tj_strcpy(char s1[], const char s2[]);	将字符串 s2 复制到 s1 中，覆盖 s1 中原内容，复制时包含\0	0
int tj_strncpy(char s1[], const char s2[], const int len);	将字符串 s2 的前 len 个字符复制到 s1 中，复制时不含\0 ★ 若 len 比 s2 的长度大，复制 s2 长度个字符即可(不含\0)	0
int tj_strcmp(const char s1[], const char s2[]);	比较字符串 s1 和 s2 的大小，英文字母要区分大小写	相等为 0，不等则为第 1 个不相等字符的 ASCII 差值

int tj_strcasecmp(const char s1[], const char s2[]);	比较字符串 s1 和 s2 的大小，英文字母不分大小写
	★ 例：tj_strcasecmp("abc", "ABZ");
	tj_strcasecmp("ABC", "abz");
	均返回-23
	★ 例：tj_strcasecmp("abZ", "AB[");
	tj_strcasecmp("abz", "AB[");
	均返回 31
		相等为 0，不等则为第 1 个不相等字符的 ASCII 差值
		★ 若不相等处字符是不同的大小写字母，则统一转换为小写后比较
		★ 若不相等处是大写字母和其它字符，则返回对应小写字母和其它字符的差值

int tj_strncmp(const char s1[], const char s2[], const int len);	比较字符串 s1 和 s2 的前 len 个字符的大小，英文字母要区分大小写 ★ 若 len 大于 s1/s2 中长度短的串，则比较到短串的\0 即结束	相等为 0，不等则为第 1 个不相等字符的 ASCII 差值
int tj_strcasencmp(const char s1[], const char s2[], const int len);	比较字符串 s1 和 s2 的前 len 个字符的大小，英文字母不分大小写 ★ 长度要求同 tj_strncmp ★ 大小写要求同 tj_strcasecmp	相等为 0，不等则为第 1 个不相等字符的 ASCII 差值

int tj_strupr(char str[]);	将字符串 str 中所有小写字母均转为大写，其它字符不变，转换后放在原串中	0
int tj_strlwr(char str[]);	将字符串 str 中所有大写字母均转为小写，其它字符不变，转换后放在原串中	0
int tj_strchr(const char str[], const char ch);	在字符串 str 中寻找字符 ch 第 1 次出现的位置，顺序是从左到右	找到：返回 1-n(位置从 1 开始)，未找到则返回 0
int tj_strstr(const char str[], const char substr[]);	在字符串 str 中寻找字符串 substr 第 1 次出现的位置，顺序是从左到右	找到：返回 1-n(位置从 1 开始)，未找到则返回 0
int tj_strrchr(const char str[], const char ch);	在字符串 str 中寻找字符 ch 第 1 次出现的位置，顺序是从右到左 ★ 例：tj_strrchr("abcdab", 'a')返回 5	找到：返回 1-n(位置从 1 开始)，未找到则返回 0
int tj_strrstr(const char str[], const char substr[]);	在字符串 str 中寻找字符串 substr 第 1 次出现的位置，顺序是从右到左 ★ 例：tj_strrstr("abcdab", "ab")返回 5	找到：返回 1-n(位置从 1 开始)，未找到则返回 0
int tj_strrev(char str[]);	字符串反转，放入原串中	0

【要求:】1、**不允许**使用任何系统函数（strlen、strcpy 等），**不允许**使用 C++的 string 类，**不允许**借助指针，**不允许**定义全局变量

2、可以用自己定义的函数（例如在其它 tj_**函数中调用 tj_strlen）

3、函数实现时不必考虑空间不够的情况（空间由调用函数保证）

4、给出 5-b12.h、5-b12-main.cpp、5-b12-sub.cpp 三个文件共同形成一个可执行文件，5-b12.h 用于函数声明，5-b12-main.cpp 是测试用例，这两个文件**不准改动，无需提交**；列表中所有函数的具体实现均在 5-b12-sub.cpp 中，每个函数实现时有具体要求，必须按要求实现，提交时只需提交此文件即可。

5、受限于目前所学知识，部分函数的返回类型、实现的具体要求与系统的 str***函数不同，**注意按文档要求实现**

13、　生成并打印 Windows 扫雷游戏的内部数组结构

【Windows 扫雷游戏的玩法:】1、开始游戏，以高级难度 16*30 的位置中 99 颗雷为例，此时虽然屏幕无显示，但 99 颗雷在什么位置内部已知

2、按下鼠标左键，表示玩家确认该位置不是雷，此时若其周围 8 个位置均无雷（**四角位置：1-3，四边位置：1-5，下同**),则屏幕显示空白(会将所有相连的空白位置全部显示),否则会按周围 8 个位置有几颗雷来显示数字 1-8；如果该位置是雷，则给出提示，游戏结束

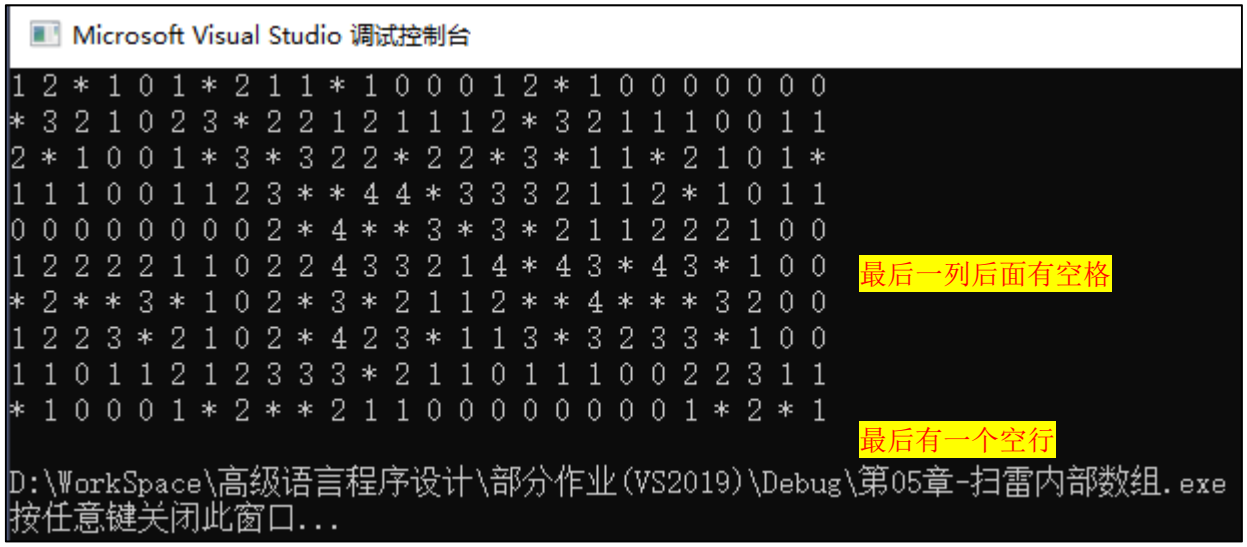
3、按下鼠标右键，表示玩家确认该位置是雷，此时屏幕会显示小红旗（如果玩家判断错误，此处不应是雷，会导致后续判断错误）

【要求:】1、在 10x26 的范围内随机产生 50 颗雷(若生成的位置已有雷，则需要再次生成新位置，即必须保证满 50 颗雷)

2、数组的大小以 10x26 为基准，如果因为程序实现需要，允许适当放大

3、其它非雷位置分别给出 0-8，表示其周围 8 个位置的雷数

4、输出形式如下，给出 5-b13-demo.exe 供参考



注：为了方便查看，要求打印时每个符号/数字间加了一个空格，包括最后一列后面

14、　从文件中读取 Windows 扫雷游戏的内部数组结构，来验证上一题的答案是否正确

【要求:】1、本题输入数据的读取方式

方式一：将上一题的输出，重定向到文件中，再将此文件做为本题的输入重定向文件

方式二：利用管道运算符将上一题的输出直接做为本题的输入，即 5-b13.exe | 5-b14.exe

【提示】1：两种方式的读入，均要在读入过程中过滤上题输出中的每个符号/数字间的空格

2：构造错误数据的方法：将上题的输出重定向文件中故意改错部分数据

- 改错数据是指*/0-8 相互错误（例：原*位置改为 2，原 0 位置改成 5）
- 不考虑改成其它字符的错误（例：将*改为#，将 2 改成 9，将空格改成 A）

2、检查条件如下

- 星号的个数是否是 50 个，不是则输出"错误 1"后程序结束
- 在*个数正确的前提下，重新计算周围的雷数，再和读入的内容进行比较，任一不匹配则输出"错误 2"后程序结束
- 通过上面两个检查则输出"正确"后程序结束

3、程序的输入不需要任何提示，输出只有一行，内容为"正确"/"错误 1"/"错误 2"，最后带一个换行符即可

4、本题需要相互验证（甲的本题去验证乙的上一题生成的数据文件），每人需要验证至少 5 人的上一题，将名单放在源程序的第 2 行用注释说明即可

```
/* 2151234 张三 计算机 */
/* 2151111 李四 2152222 王五 2153333 赵六 ...*/
```

- 两人之间必须双向，即甲乙的 5-b14 互验对方的 5-b13
- 正常情况双向查验都应该正确，任一查验不正确则同步扣分
- 查验的名单必须放在第二行，且多人的信息必须写在一行内，不要换行。信息按学号 姓名的格式依次排列即可，中间用空格分隔，不要加其它多余的字符，注释符单行/多行均可（信息格式写错则认为未相互验证）

【编译器要求:】

		编译器VS	编译器Dev	编译器Linux
5-b11-1.c	人民币大写(char数组,一句输出,C方式)	Y	Y	Y
5-b11-2.cpp	人民币大写(string类,一句输出)	Y	Y	Y
5-b12-sub.cpp	一组字符串处理函数	Y	Y	Y
5-b13.cpp	扫雷内部数组生成	Y	Y	Y
5-b14.c	扫雷内部数组验证(C方式)	Y	Y	Y

注：Linux 仅限计算机拔尖班（10069201/10071701）同学，其它班级忽略即可

【作业要求：】

1、11 月 25 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会自动扣除相应的分数，具体见网页上的说明

4、另：本周还有**综合题**下发，请自行合理安排时间