补充练习1：

数组：

1. 矩阵乘方

#include<stdio.h>

#include<math.h>

int a[15][15];

int main(){

int n, b, m;

scanf("%d%d%d", &n, &b, &m);

puts("输入矩阵：");

for(int i = 0; i < n; ++ i)

for(int j = 0; j < n; ++ j)

scanf("%d", &a[i][j]);

puts("转换结果为：");

for(int i = 0; i < n; ++ i){

for(int j = 0; j < n; ++ j){

printf("%d ", (int)pow(a[i][j], b) % m);

}

puts("");

}

return 0;

}

用math库里的pow函数完成乘方的运算。但是pow会把数据类型由整型变成浮点型double，这样就不能完成取模运算了，所以还得强制类型转换，把浮点型转回整型。



补充练习2：

字符串

1. 倒序输出

#include<stdio.h>

#include<string.h>

//char tmp[15]; //定义全局变量就不用过多考虑传参了

char \*rev(char \*s, char \*tmp){

int n = strlen(s);

int j = 0;

for(int i = n - 1; i >= 0; -- i, ++ j)

tmp[j] = s[i];

tmp[j] = '\0'; //休止符容易遗漏

return tmp;

}

int main(){

char s[15], tmp[15];

puts("输入字符串：");

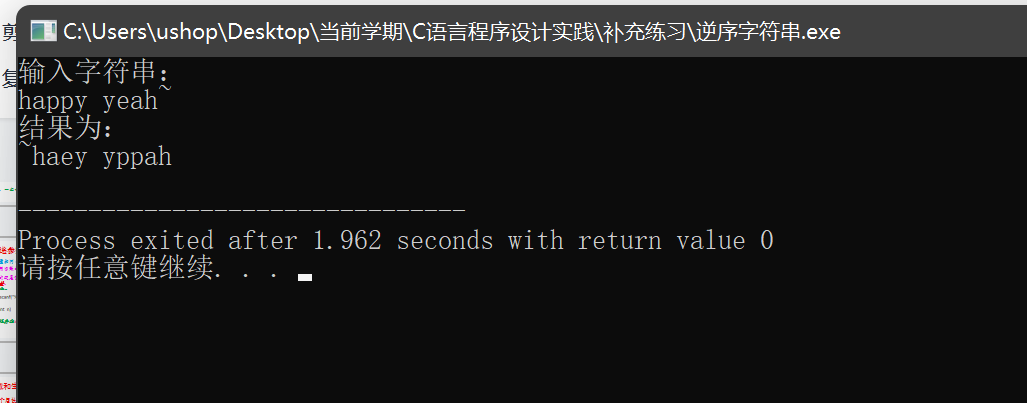
gets(s);

puts("结果为：");

puts(rev(s, tmp));

return 0;

}



1. 顺序输出元音字符

#include<stdio.h>

#include<string.h>

const char str[11] = "AEIOUaeiou"; //定义常量字符串 用循环遍历 判断更方便

char \*chg(char \*s, char \*tmp) {

int n = strlen(s);

int j = 0;

for(int i = 0; i < n; ++ i) {

for(int k = 0; k < 10; ++ k) {

if(s[i] == str[k]) {

tmp[j ++] = s[i]; //如果是元音则存入tmp字符数组

break;

}

}

}

tmp[j] = '\0';

return tmp;

}

int main() {

char s[15], tmp[15];

puts("输入字符串：");

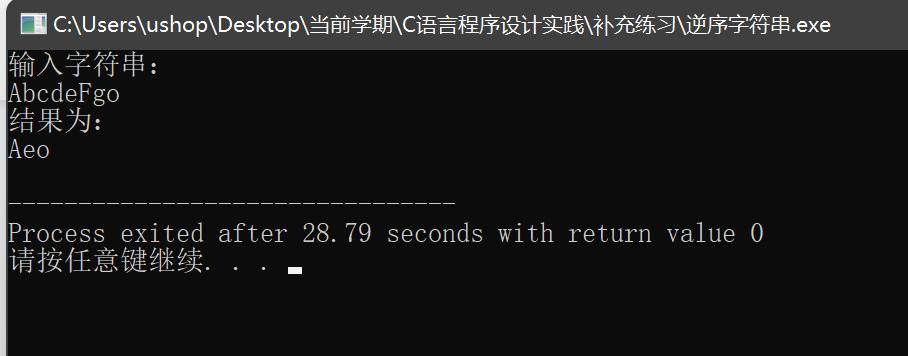
gets(s);

puts("结果为：");

puts(chg(s, tmp));

return 0;

}



补充练习3

人工解数独题：

（这是降低难度版，只能解决具有唯一解的数独问题，为了方便用了c++语言）

#include <cstdio>

#include <cstring>

using namespace std;

const int N = 10;

// 由于是直接用字符数组存数独，以字符串的方式读入，每行后面在读入的时候会被自动加上一个 '\0'，所以数组大小 N 要开到 10，才能避免溢出。

char g[N][N];

bool dfs(int x, int y) {

if (y == 9) return true; // 如果当前列跳出了最后一列，则直接放回 true

if (x == 9) return dfs(0, y + 1); // 如果当前行跳出了最后一行，则返回下一列第?行

if (g[x][y] != '.') return dfs(x + 1, y); // 如果当前行已有数字，直接跳过

bool st[N]; // st 数组存当前位置 (x, y) 还能填哪些数

memset(st, false, sizeof st); // 要记得初始化~

for (int i = 0; i < N - 1; i ++ ) // 看一下该列上有哪些数字被填过了

if (g[i][y] > 47 && g[i][y] < 58)

st[g[i][y] ^ 48] = true;

for (int i = 0; i < N - 1; i ++ ) // 看一下该行上有哪些数字被填过了

if (g[x][i] > 47 && g[x][i] < 58)

st[g[x][i] ^ 48] = true;

int sx = x / 3 \* 3, sy = y / 3 \* 3; // 找到当前九宫格的左上角位置

for (int i = sx; i < sx + 3; i ++ ) // 看一下该九宫格内有哪些数字被填过了

for (int j = sy; j < sy + 3; j ++ )

if (g[i][j] > 47 && g[i][j] < 58)

st[g[i][j] ^ 48] = true;

for (int i = 1; i < N; i ++ ) // 枚举当前格内能填的所有数字

if (!st[i]) {

g[x][y] = i ^ 48; // 如果能填，那么填上，并搜索下一格

if (dfs(x + 1, y)) return true;

}

g[x][y] = '.'; // 如果搜完了所有可填数字，或没有可填数字，那么将该格改为未填状态

return false; // 并返回 false

}

int main() {

for (int i = 0; i < N - 1; i ++ )

scanf("%s", g[i]); // 以字符串方式读入

dfs(0, 0); // 从 (0, 0) 开始爆搜

for (int i = 0; i < N - 1; i ++) {

putchar('\n');

for (int j = 0; j < N - 1; j ++ )

putchar(g[i][j]); // 输出填好的数独。由于数据保证有解且有唯一解，所以不需特判任何情况，直接输出即可

}

return 0;

}

