1.

#define \_CRT\_SECURE\_NO\_WARNINGS

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

int main(int argc, char \*argv[]) /\*argv[1]：输入文件名；argv[2]：输出文件名\*/

{

char \*\*s;

int size = 0, capacity = 4; /\*字符串数目与容量\*/

int range; /\*供排序\*/

FILE \*fin, \*fout;

if (argc != 3)

{

printf("Parameter input error!\n");

exit(-1);

}

s = (char \*\*)malloc(sizeof(char \*) \* capacity); /\*定义字符串数组\*/

if (!s)

{

perror("ERROR!\n");

exit(-1);

}

for (int i = 0; i < capacity; ++i) /\*分配capacity个字符串空间\*/

{

if (!(s[i] = (char \*)malloc(sizeof(char) \* 82)))

{

perror("ERROR!\n");

exit(-1);

}

}

fin = fopen(argv[1], "r"); /\*读入文件\*/

if (!fin)

{

printf("Cannot open the file %s!\n", argv[1]);

exit(-1);

}

fout = fopen(argv[2], "w"); /\*读出文件\*/

if (!fout)

{

printf("Cannot open the file %s!\n", argv[2]);

exit(-1);

}

while (1)

{

if (!(fgets(s[size++], 82, fin)))

break;

if (size == capacity) /\*达到最大值，重新分配内存\*/

{

capacity \*= 2;

s = (char \*\*)realloc(s, capacity \* sizeof(char\*));

if (!s)

{

perror("ERROR!\n");

exit(-1);

}

for (int i = capacity / 2; i < capacity; ++i)

{

if (!(s[i] = (char \*)malloc(sizeof(char) \* 82)))

{

perror("ERROR!\n");

exit(-1);

}

}

}

}

--size;

range = strlen(s[size - 1]); /\*考察最后一个字符串末尾是否有换行符\*/

if (s[size - 1][range - 1] != '\n')

{

s[size - 1][range] = '\n';

s[size - 1][range + 1] = '\0';

}

range = size - 1;

while (range > 0) /\*排序字符串\*/

{

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

{

if (strcmp(s[i], s[i + 1]) > 0)

{

char \*tmp = s[i];

s[i] = s[i + 1];

s[i + 1] = tmp;

}

}

--range;

}

for (int i = 0; i < size; ++i) /\*输出字符串\*/

fprintf(fout, s[i]);

for (int i = capacity - 1; i >= 0; --i) /\*释放内存\*/

free(s[i]);

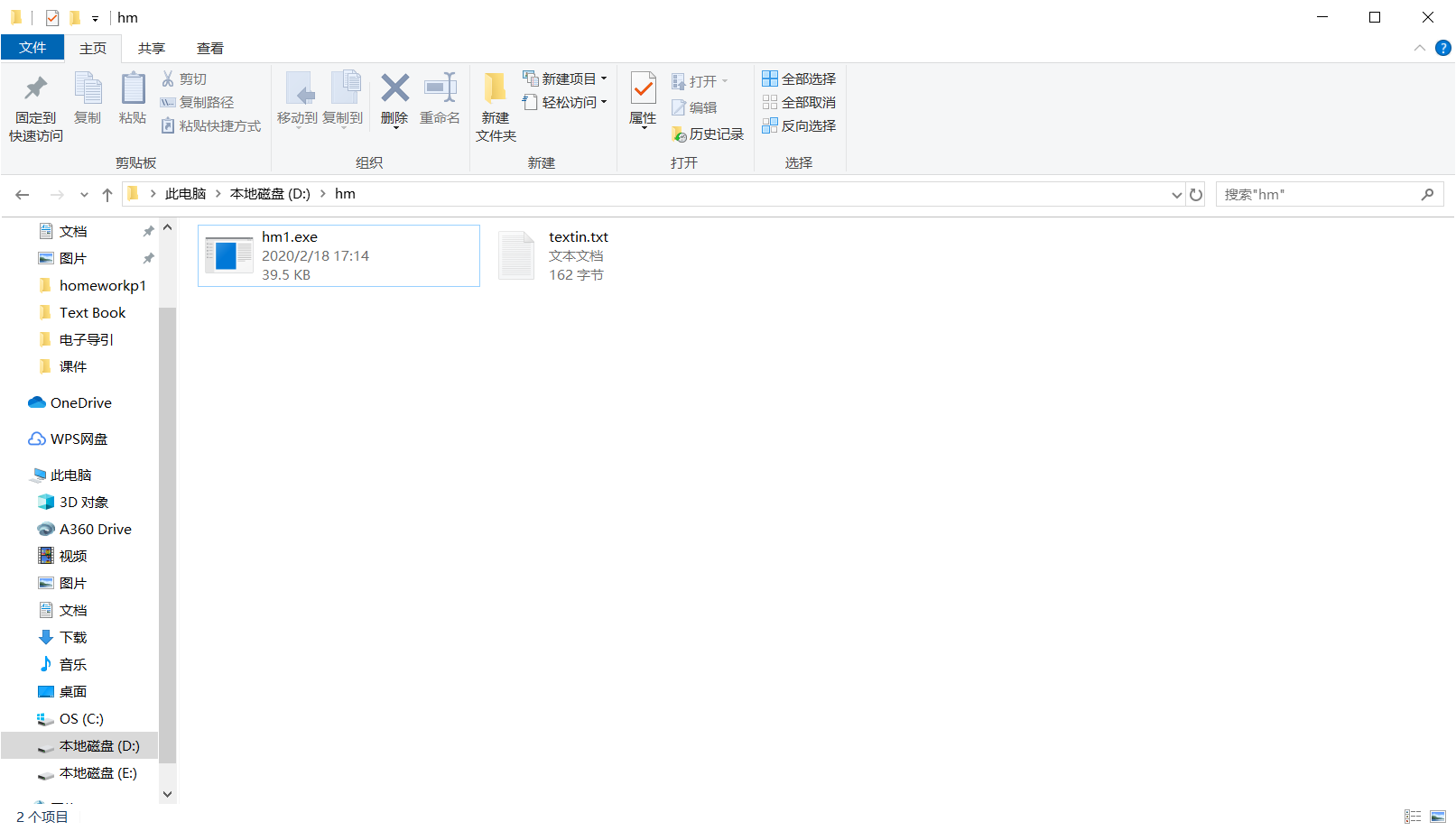
free(s);

fclose(fout); /\*关闭文件\*/

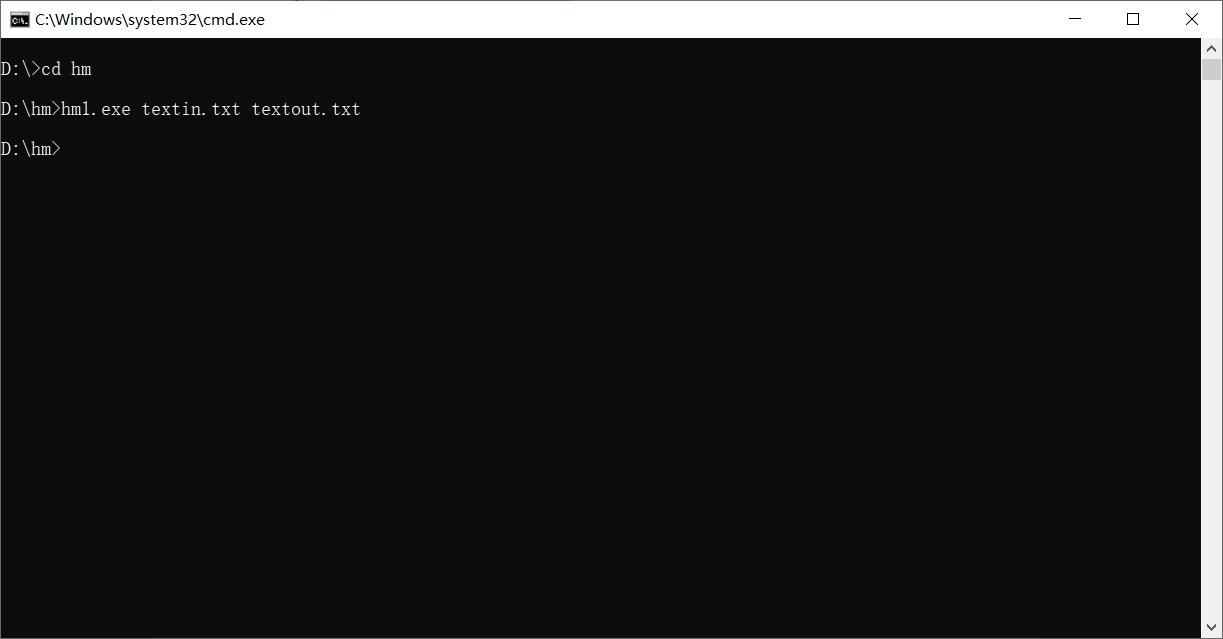
fclose(fin);

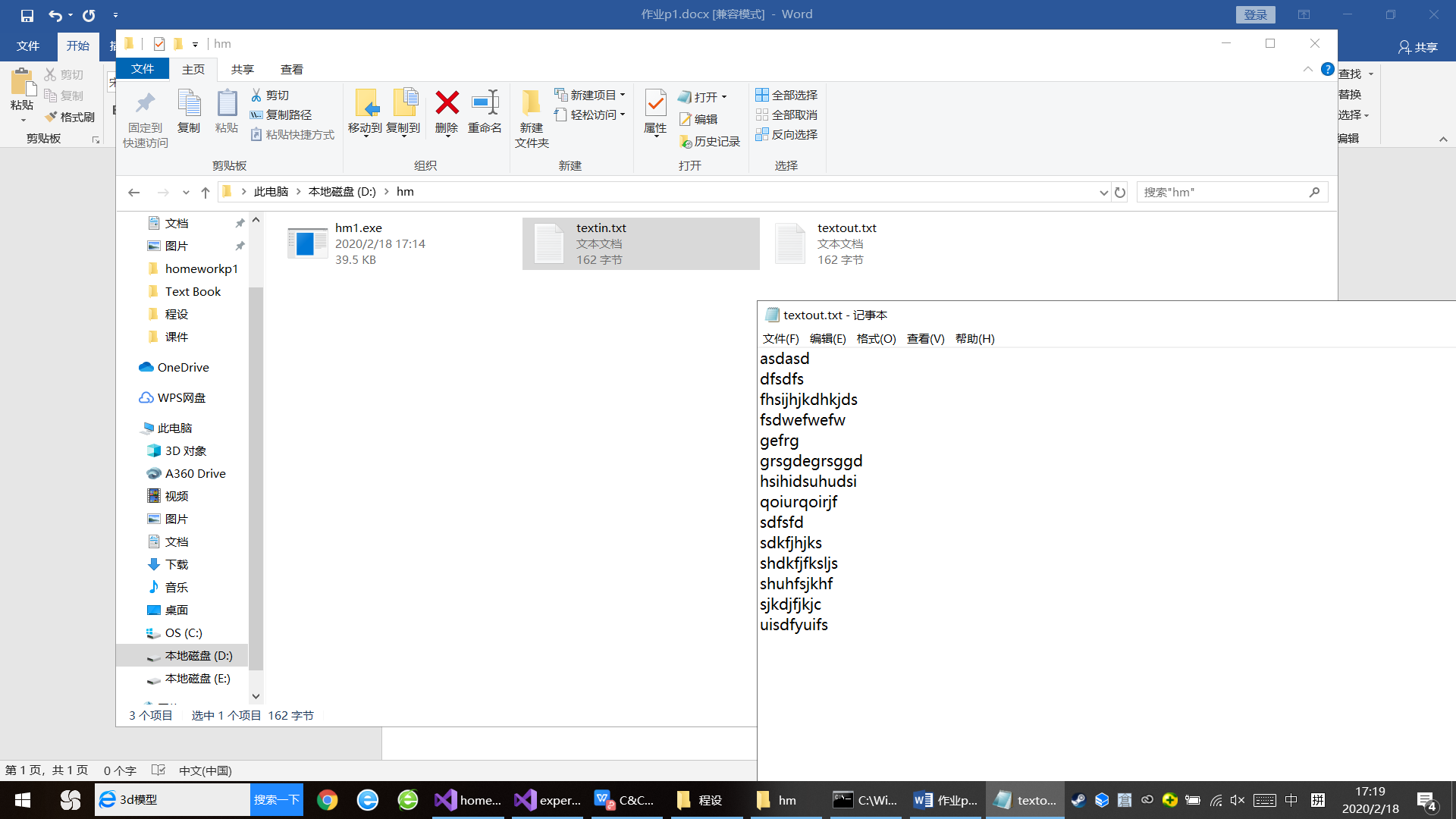
return 0;

}









2.

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#pragma warning(disable:4996)

typedef unsigned char UCHAR;

inline UCHAR rev(UCHAR x) /\*反转\*/

{

x = (((x & 0x0f) << 4) | ((x & 0xf0) >> 4)); /\*相邻四位反转\*/

x = (((x & 0x33) << 2) | ((x & 0xcc) >> 2)); /\*相邻两位反转\*/

x = (((x & 0x55) << 1) | ((x & 0xaa) >> 1)); /\*相邻一位反转\*/

return x;

}

int main(int argc, char\* argv[]) /\*argv[1]：输入文件名；argv[2]：输出文件名\*/

{

UCHAR form[256]; /\*存储反转值\*/

UCHAR\* buffer;

FILE\* fin, \* fout;

int max = 16;

size\_t now;

if (argc != 3)

{

printf("Parameter input error!\n");

exit(-1);

}

for (register int i = 0; i < 256; ++i)

{

form[i] = rev((UCHAR)i);

}

if (!(fin = fopen(argv[1], "rb")))

{

printf("Cannot open the file %s!\n", argv[1]);

exit(-1);

}

if (!(fout = fopen(argv[2], "wb")))

{

printf("Cannot open the file %s!\n", argv[2]);

exit(-1);

}

for (int i = 0; i < 256; ++i) /\*初始化反转表\*/

form[i] = rev(i);

buffer = (UCHAR\*)malloc(sizeof(UCHAR) \* max);

if (!buffer)

{

printf("ERROR!");

exit(-1);

}

while (1)

{

now = fread(buffer, sizeof(UCHAR), max, fin);

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

buffer[i] = form[buffer[i]];

fwrite(buffer, sizeof(UCHAR), now, fout);

if (now < max) break; /\*到达文件尾\*/

free(buffer); /\*增大每次读入量\*/

max \*= 2;

buffer = (UCHAR\*)malloc(sizeof(UCHAR) \* max);

if (!buffer)

{

printf("ERROR!");

exit(-1);

}

}

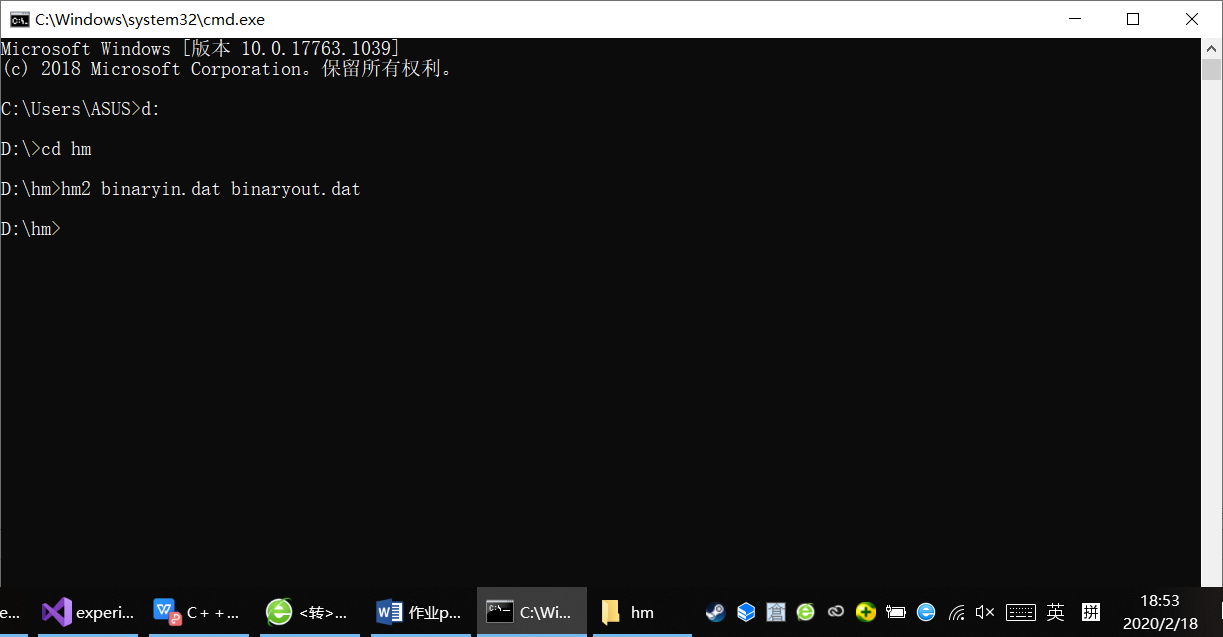
free(buffer);

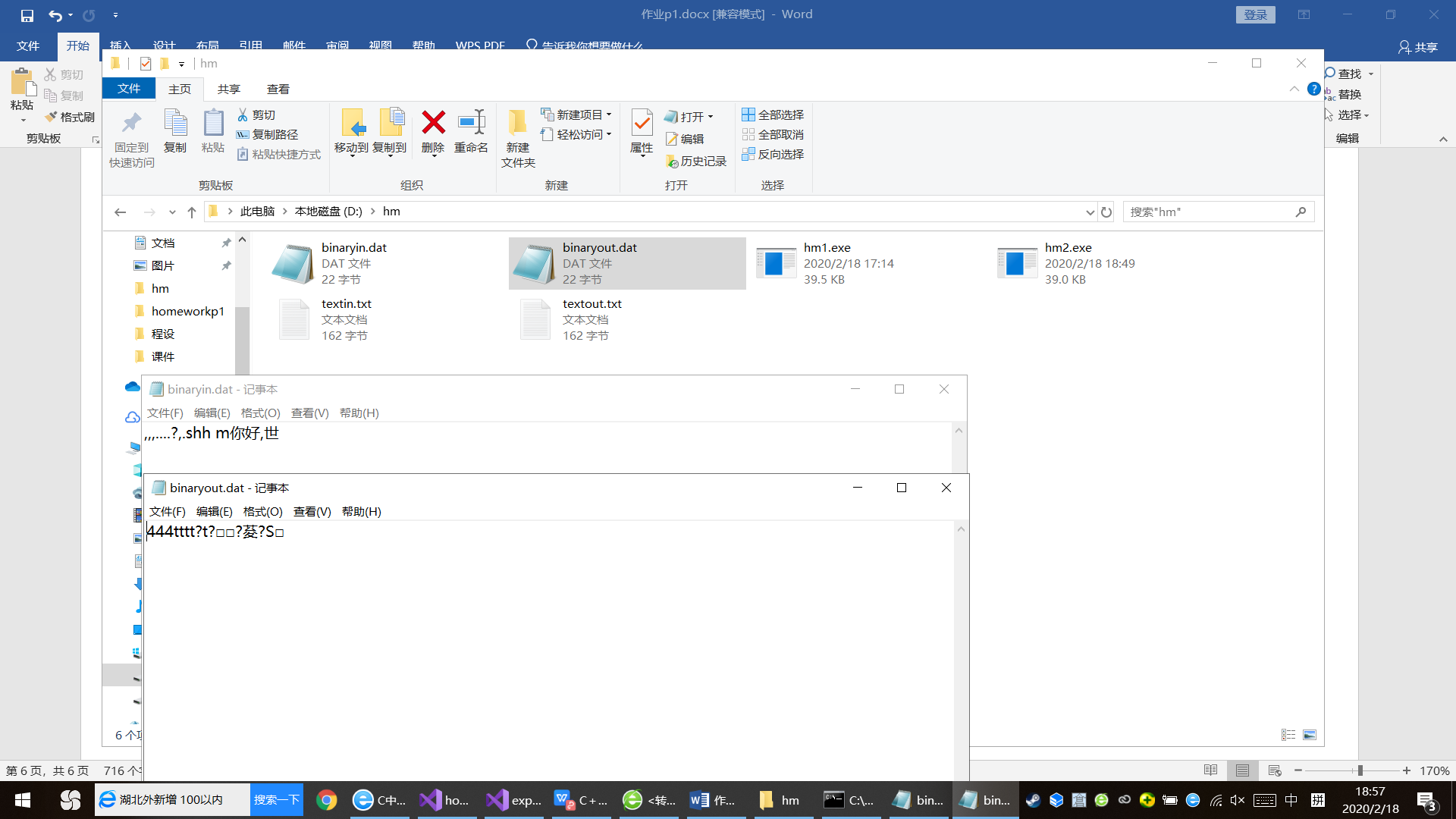
fclose(fout);

fclose(fin);

return 0;

}





3. 探究题：

#include <stdio.h>

#include <string.h>

#include <stdlib.h>

#pragma warning(disable:4996)

char input[20000] = { 0 };

char output[20000];

char number[10][3] = { "零", "一", "二", "三", "四" , "五" , "六" , "七" , "八" , "九" };

char loword[4][3] = { "个", "十", "百", "千" };

char highword[13][3] = { "个", "万", "亿", "兆", "京", "垓", "秭", "穰", "沟", "涧", "正", "载", "极" };

#define MAX\_NUM 52

inline long changeToNum(char\* p) //将双字节字符转换成ANSI码值

{

return (unsigned char)p[0] \* 0x100L + (unsigned char)p[1];

}

int main()

{

char zero[] = "０";

char nine[] = "９";

char\* p = input, \* q, \* tmptr;

int outlen = 0, pN;

int zeroN, nineN, tmp, tmp1, tmp2;

int tmp3, zeronum, tmp4;

//char outputzero = 0; //刚才是否输出了'零'

zeroN = changeToNum(zero);

nineN = changeToNum(nine);

printf("请输入一段（不含空格、回车、Tab！！！！）的话：\n\n");

scanf("%s", input);

while (\*p)

{

if (\*p == '-' && ((\*(p + 1) >= '0' && \*(p + 1) <= '9') || (changeToNum(p + 1) >= zeroN && changeToNum(p + 1) <= nineN)))

{

strncpy(&output[outlen], "负", 2);

outlen += 2;

++p;

}

if (((\*p) >> 7) & 1) //最高位为1

{

/\*处理全角\*/

pN = changeToNum(p);

if (pN >= zeroN && pN <= nineN)

{

if (pN == zeroN) //处理开头的0

{

p += 2;

while (changeToNum(p) == zeroN) { p += 2; } //去掉开头的多个零，只保留一个

p -= 2;

if (changeToNum(p + 2) < zeroN || changeToNum(p + 2) > nineN) //整数部分是零

{

strncpy(&output[outlen], "零", 2);

outlen += 2;

p += 2;

goto dot1;

}

p += 2;

}

if (changeToNum(p) == zeroN + 1 && (changeToNum(p + 4) < zeroN || changeToNum(p + 4) > nineN)) //10比较特殊，因为“1”不读

{

if (changeToNum(p + 2) >= zeroN && changeToNum(p + 2) <= nineN)

{

strncpy(&output[outlen], "十", 2);

outlen += 2;

if (changeToNum(p + 2) != zeroN)

{

strncpy(&output[outlen], number[changeToNum(p + 2) - zeroN], 2);

outlen += 2;

}

p += 4;

goto dot1;

}

}

q = p + 2;

while (changeToNum(q) >= zeroN && changeToNum(q) <= nineN)

{

q += 2;

}

tmp = (q - p) / 2;

if (tmp > MAX\_NUM)

{

perror("\n这段话里包含过大的数字！\n");

exit(-1);

}

tmp -= 1;

tmp1 = tmp / 4;

tmp2 = tmp % 4;

if (tmp2 == 1 && changeToNum(p) == zeroN + 1 && changeToNum(p + 2) >= zeroN && changeToNum(p + 2) <= nineN) //十万、十亿等特殊，1不读

{

strncpy(&output[outlen], "十", 2);

outlen += 2;

if (changeToNum(p + 2) != zeroN)

{

strncpy(&output[outlen], number[changeToNum(p + 2) - zeroN], 2);

outlen += 2;

}

strncpy(&output[outlen], highword[tmp1], 2);

outlen += 2;

tmp2 = 3;

--tmp1;

}

while (tmp1 >= 0)

{

//遇到0

if (changeToNum(&p[(tmp - 4 \* tmp1 - tmp2) \* 2]) == zeroN)

{

tmp3 = tmp - 4 \* tmp1 - tmp2 + 1;

zeronum = 1;

while (changeToNum(&p[tmp3 \* 2]) == zeroN && tmp2 - zeronum >= 0)

{

++zeronum;

++tmp3;

}

if (zeronum == 4) //四位都是0

{

tmp4 = 0;

while (changeToNum(&p[(tmp3 + tmp4) \* 2]) == zeroN)

{

++tmp4;

}

if (tmp4 == tmp1 \* 4) //0延续到个位，直接判断小数点

{

p = q;

goto dot1;

}

//if (!outputzero) //刚刚没有输出'零'则输出'零'

{

strncpy(&output[outlen], number[0], 2);

outlen += 2;

//outputzero = 1;

}

tmp1 -= tmp4 / 4 + 1; //跳转到非零处

tmp2 = 3 - tmp4 % 4;

continue;

}

if (zeronum == tmp2 + 1) /\*0延续到每个4位末尾\*/

{

if (tmp1 > 0) //需要输出高级计数单位

{

strncpy(&output[outlen], highword[tmp1], 2);

outlen += 2;

//outputzero = 0;

}

--tmp1;

tmp2 = 3;

continue;

}

//if (!outputzero) //刚刚没有输出'零'

{

strncpy(&output[outlen], number[0], 2);

outlen += 2;

//outputzero = 1;

}

tmp2 -= zeronum;

continue;

}

//非零

//数字

strncpy(&output[outlen], number[changeToNum(p + (tmp - 4 \* tmp1 - tmp2) \* 2) - zeroN], 2);

outlen += 2;

//outputzero = 0;

if (tmp2 != 0) //不是个位

{ //位

strncpy(&output[outlen], loword[tmp2], 2);

outlen += 2;

}

--tmp2;

if (tmp2 == -1)

{

if (tmp1 != 0)

{

strncpy(&output[outlen], highword[tmp1], 2);

outlen += 2;

}

--tmp1;

tmp2 = 3;

}

}

p = q;

dot1:

//处理小数点

if (\*p == '.')

{

++p;

if (changeToNum(p) >= zeroN && changeToNum(p) <= nineN)

{

strncpy(&output[outlen], "点", 2);

outlen += 2;

strncpy(&output[outlen], number[changeToNum(p) - zeroN], 2);

outlen += 2;

p += 2;

}

while (changeToNum(p) >= zeroN && changeToNum(p) <= nineN)

{

strncpy(&output[outlen], number[changeToNum(p) - zeroN], 2);

outlen += 2;

p += 2;

}

}

}

else

{

output[outlen++] = \*(p++);

output[outlen++] = \*(p++);

//outputzero = 0;

}

//int pN = changeToNum(p);

//if (pN >= zeroN && pN <= nineN)

//{

// /\*转换成字符串\*/

// /\*拷贝进输出串\*/

// /\*修改p\*/

//}

//else

//{

//}

}

else

{

/\*if (\*p == '-' && \*(p + 1) >= '0' && \*(p + 1) <= '9')

{

strncpy(&output[outlen], "负", 2);

outlen += 2;

++p;

}\*/

if (\*p >= '0' && \*p <= '9')

{

if (\*p == '0') //处理开头的0

{

++p;

while (\*p == '0') { ++p; } //去掉开头的多个零，只保留一个

--p;

if (\*(p + 1) < '0' || \*(p + 1) > '9') //整数部分是零

{

strncpy(&output[outlen], "零", 2);

outlen += 2;

++p;

goto dot;

}

++p;

}

if (\*p == '1' && (\*(p + 2) < '0' || \*(p + 2) > '9')) //10比较特殊，因为“1”不读

{

if (\*(p + 1) >= '0' && \*(p + 1) <= '9')

{

strncpy(&output[outlen], "十", 2);

outlen += 2;

if (\*(p + 1) != '0')

{

strncpy(&output[outlen], number[\*(p + 1) - '0'], 2);

outlen += 2;

}

p += 2;

goto dot;

}

}

q = p + 1;

while (\*q >= '0' && \*q <= '9')

{

++q;

}

tmp = q - p;

if (tmp > MAX\_NUM)

{

perror("\n这段话里包含过大的数字！\n");

exit(-1);

}

--tmp;

tmp1 = tmp / 4;

tmp2 = tmp % 4;

if (tmp2 == 1 && \*p == '1' && \*(p + 1) >= '0' && \*(p + 1) <= '9') //十万、十亿等特殊，1不读

{

strncpy(&output[outlen], "十", 2);

outlen += 2;

if (\*(p + 1) != '0')

{

strncpy(&output[outlen], number[\*(p + 1) - '0'], 2);

outlen += 2;

}

strncpy(&output[outlen], highword[tmp1], 2);

outlen += 2;

tmp2 = 3;

--tmp1;

}

while (tmp1 >= 0)

{

//遇到0

if (p[tmp - 4 \* tmp1 - tmp2] == '0')

{

tmp3 = tmp - 4 \* tmp1 - tmp2 + 1;

zeronum = 1;

while (p[tmp3] == '0' && tmp2 - zeronum >= 0)

{

++zeronum;

++tmp3;

}

if (zeronum == 4) //四位都是0

{

tmp4 = 0;

while (p[tmp3 + tmp4] == '0') ++tmp4;

if (tmp4 == tmp1 \* 4) //0延续到个位，直接判断小数点

{

p = q;

goto dot;

}

//if (!outputzero) //刚刚没有输出'零'则输出'零'

{

strncpy(&output[outlen], number[0], 2);

outlen += 2;

//outputzero = 1;

}

tmp1 -= tmp4 / 4 + 1; //跳转到非零处

tmp2 = 3 - tmp4 % 4;

continue;

}

if (zeronum == tmp2 + 1) /\*0延续到每个4位末尾\*/

{

if (tmp1 > 0) //需要输出高级计数单位

{

strncpy(&output[outlen], highword[tmp1], 2);

outlen += 2;

//outputzero = 0;

}

--tmp1;

tmp2 = 3;

continue;

}

//if (!outputzero) //刚刚没有输出'零'

{

strncpy(&output[outlen], number[0], 2);

outlen += 2;

//outputzero = 1;

}

tmp2 -= zeronum;

continue;

}

//非零

//数字

strncpy(&output[outlen], number[p[tmp - 4 \* tmp1 - tmp2] - '0'], 2);

outlen += 2;

//outputzero = 0;

if(tmp2 != 0) //不是个位

{ //位

strncpy(&output[outlen], loword[tmp2], 2);

outlen += 2;

}

--tmp2;

if (tmp2 == -1)

{

if (tmp1 != 0)

{

strncpy(&output[outlen], highword[tmp1], 2);

outlen += 2;

}

--tmp1;

tmp2 = 3;

}

}

p = q;

dot:

//处理小数点

if (\*p == '.')

{

++p;

if (\*p >= '0' && \*p <= '9')

{

strncpy(&output[outlen], "点", 2);

outlen += 2;

strncpy(&output[outlen], number[\*p - '0'], 2);

outlen += 2;

++p;

}

while (\*p >= '0' && \*p <= '9')

{

strncpy(&output[outlen], number[\*p - '0'], 2);

outlen += 2;

++p;

}

}

}

else

{

output[outlen++] = \*(p++);

//outputzero = 0;

}

}

}

output[outlen++] = '\0';

printf("\n\n%s\n", output);

return 0;

}

运行结果如下：

