

The background features a stylized tree with a blue trunk and branches, adorned with green, grey, and yellow hexagonal leaves. Large, semi-transparent circles in shades of blue, green, and orange are scattered across the slide.

Huffman编/译码器 设计与实现

课程设计汇报

信息院 网络工程161班
陈扬 19316117



目录 CONTENTS

01

算法设计思想

02

程序结构

03

实验结果与分析

04

总结分析

01

算法设计思想

- Huffman树
- Huffman编码
- Huffman译码
- huf文件编码算法
- huf文件译码算法

02

程序结构

- 程序实现功能
- 程序执行流程
- 程序执行逻辑
- 函数功能

03

实验结果与分析

- 程序执行结果
- 结果分析
- 文件压缩率
- 文件译码准确率

04

总结分析

- 程序设计评价
- Huffman编码/译码
技术评价
- 自我评价

算法设计思想

- 1 Huffman树
- 2 Huffman编码
- 3 Huffman译码
- 4 huf文件编码算法
- 5 huf文件译码算法



PART 1

路径：树中一个结点到另一个结点之间的分支构成这两个结点之间的路径。

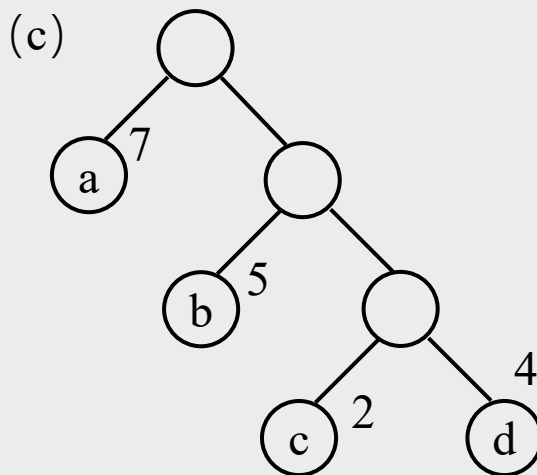
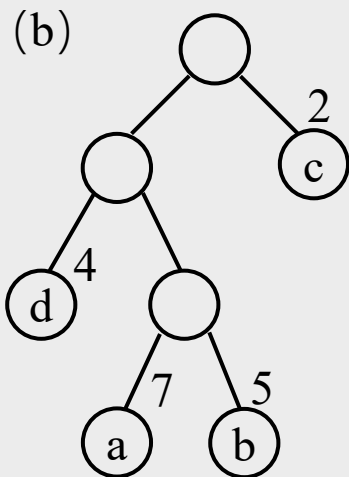
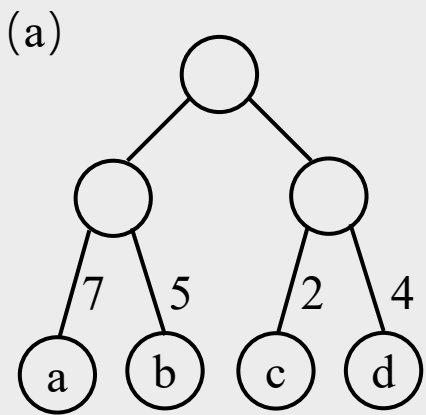
路径长度：路径上的分支数目称作路径长度。

树的路径长度：从树根到每一个结点的路径长度之和。

结点的带权路径长度：在一棵树中，如果其结点上附带有一个权值，通常把该结点的路径长度与该结点上的权值的乘机称为节点的带权路径长度。

树的带权路径长度：树中所有叶子结点的带权路径长度之和，通常记作 $WPL = \sum_{k=1}^n w_k l_k$

假设有 n 个权值 $\{w_1, w_2, \dots, w_n\}$ ，试构造一颗有 n 个叶子结点的二叉树，每个叶子结点带权为 w_i ，则其中带权路径长度 WPL 最小的二叉树称做最优二叉树或Huffman树。



(a) WPL

$$=7 \times 2 + 5 \times 2 + 2 \times 2 + 4 \times 2 \\ =36$$

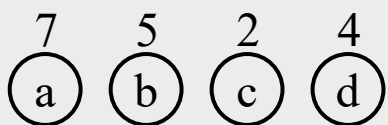
(b) WPL

$$=7 \times 3 + 5 \times 3 + 2 \times 1 + 4 \times 2 \\ =46$$

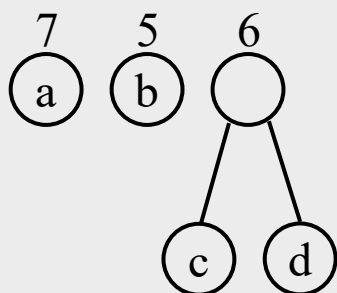
(c) WPL

$$=7 \times 1 + 5 \times 2 + 2 \times 3 + 4 \times 3 \\ =35$$

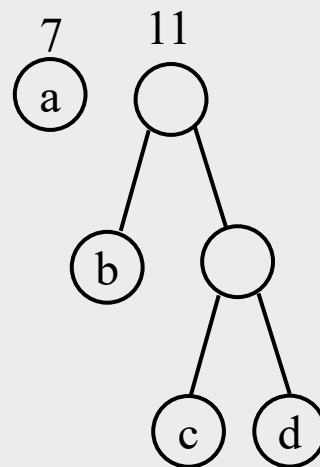
- (1) 根据给定的 n 个权值 $\{w_1, w_2, \dots, w_n\}$ 构成 n 棵二叉树的集合 $F=\{T_1, T_2, \dots, T_n\}$, 其中每棵二叉树 T_i 中只有一个带权为 w_i 的根节点, 其左右子树均空。
- (2) 在 F 中选取两棵根节点的权值最小的树作为左右子树构造一棵新的二叉树, 且置新的二叉树的根节点的权值为其左、右子树上根节点的权值之和。
- (3) 在 F 中删除这两棵树, 同时将新得到的二叉树加入 F 中。
- (4) 重复 (2) 和 (3), 直到 F 只含一棵树为止。这棵树便是Huffman树。



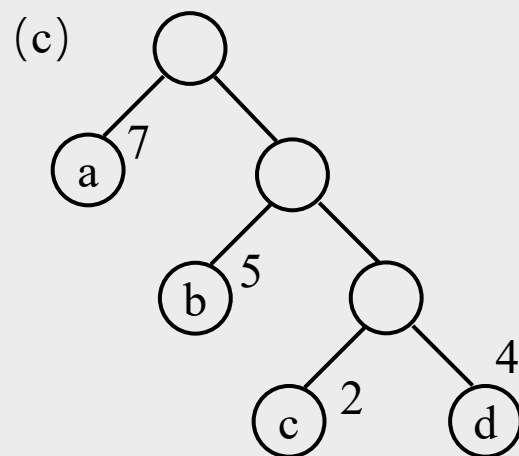
(a)



(b)



(c)



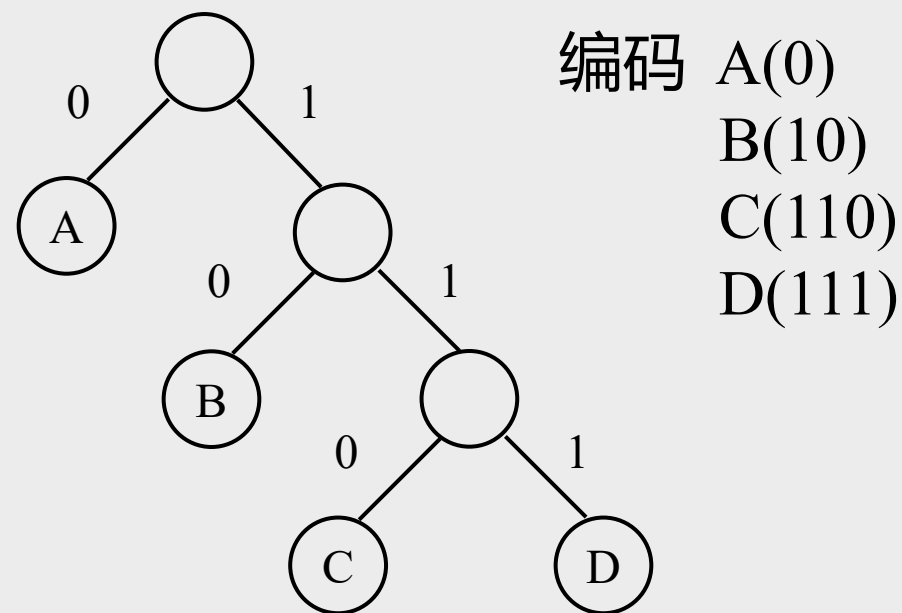
(d)

目前，进行快速远距离通信的主要手段是电报，即将需传送的文字转换成由二进制的字符组成的字符串。在设计编码时需要遵守两个原则：

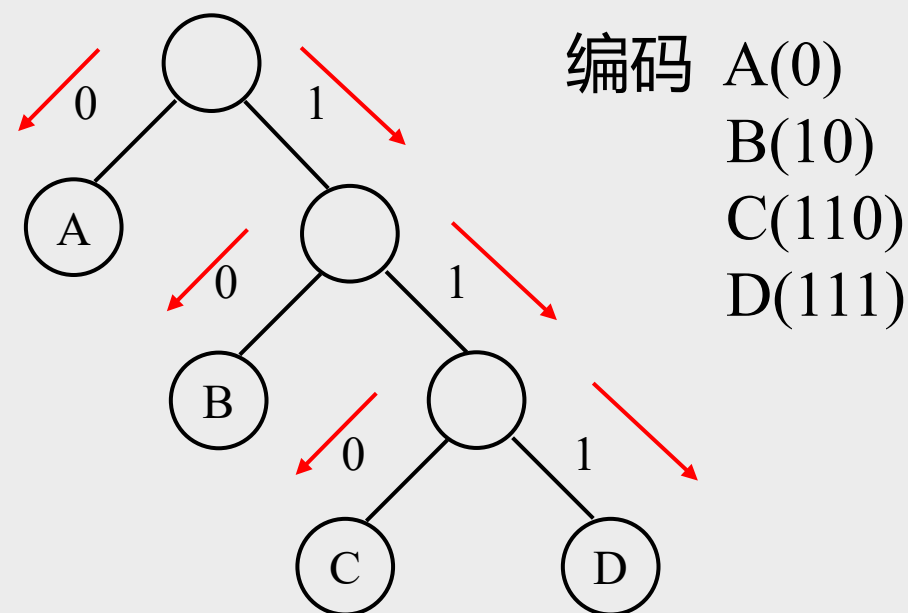
- (1) 发送方传输的二进制编码，到接收方解码后必须具有唯一性，即解码结果与发送方发送的电文完全一样；
- (2) 发送的二进制编码尽可能地短。

要设计长短不等的编码，其必须是任一个字符的编码都不是另一个字符的编码的前缀，这种编码称做**前缀编码**。

可以利用二叉树来设计二进制的前缀编码。假设有一棵如图所示的二叉树，其4个叶子结点分别表示A、B、C、D这4个字符，且约定左分支表示字符'0'，右分支表示字符'1'，则可以从根节点到叶子结点的路径上分支字符组成的字符串作为该叶子结点字符的编码。可以证明，如此得到的必为二进制前缀编码。如图所得A、B、C、D的二进制前缀编码分别为0、10、110和111。



从根结点出发，逐个读入电文中的二进制代码；若代码为0则走向左孩子，否则走向右孩子；一旦到达叶子结点，便可译出代码所对应的字符。然后又重新从根结点开始继续译码，直到二进制电文结束。



在Windows操作系统中，二进制以8位（1字节）为单位存储，8位二进制无符号数的表示范围为 $2^0-1 \sim 2^7-1$ ，即0~255。而0~255能够恰好对应ASCII码对照表中的所有256个字符，因此，在压缩存储Huffman编码时，每一次可以取8位0/1编码，将其对应到一个字节的每一个bit位上，再将这个字节转换成ASCII码值，在文件中写入其对应的字符。

ABCD ABCD ABCD \longrightarrow 010110111010110111010110111

0	1	0	1	1	0	1	1
---	---	---	---	---	---	---	---

 \longrightarrow 91 \longrightarrow [

需要考虑Huffman编码总个数不是8的整数倍的情况。

- 是，则直接进行编码
- 不是，则计算在文件最后，需要补1的个数，使得文档长度正好为8的整数倍
假设 n 为Huffman编码总长度， m 为需要补1的个数，则有 $m=8-n\%8$ 。

为了保证huf文件能够准确译出，所以在huf文件的二进制编码中，利用其第一个字节（前8位）来存储补1的个数。

在进行转换时，需要用到 $\&$ 、 $|$ 、 \sim 、 \wedge 、 \ll 、 \gg 运算，实现对字节的位操作。

```
char bit_number_transform(int n[]) {  
    char num ;  
    int i;  
    for (i = 0; i < 8; i++) {  
        if (n[i] == 1)  
            num |= (1 << (7 - i));  
        else  
            num &= ~(1 << (7 - i));  
    }  
    return num;  
}
```

在进行Huffman编码解压缩时，每一次取一个字节的字符，先将其转换成对应的ASCII码，再将ASCII码的8位二进制位'0'/'1'依次输出即可得到原来的Huffman编码。

[\longrightarrow 91 \longrightarrow

0	1	0	1	1	0	1	1
---	---	---	---	---	---	---	---

设huf文件的大小为 m 字节，翻译的到的补1个数为 p ，则有原Huffman编码长度 $n = 8 \times (m - 1) - p$

在进行转换时，需要用到 $\&$ 、 $|$ 、 \sim 、 \wedge 、 \ll 、 \gg 运算，实现对字节的位操作。

```
while (!feof(huf)) {  
    c = fgetc(huf);  
    for (t = 0; t < 8; t++) {  
        if (((c >> (7 - t)) & 1) == 1) {  
            count++;  
            fputc(49, unzip);  
            if (count == (8 * (huflength - 1) - code_mod))  
                break;  
        }  
        else if (((c >> (7 - t)) & 1) == 0) {  
            count++;  
            fputc(48, unzip);  
            if (count == (8 * (huflength - 1) - code_mod))  
                break;  
        }  
    }  
    if (count == (8 * (huflength - 1) - code_mod))  
        break;  
}
```

程序结构

1 程序实现功能

2 程序执行流程

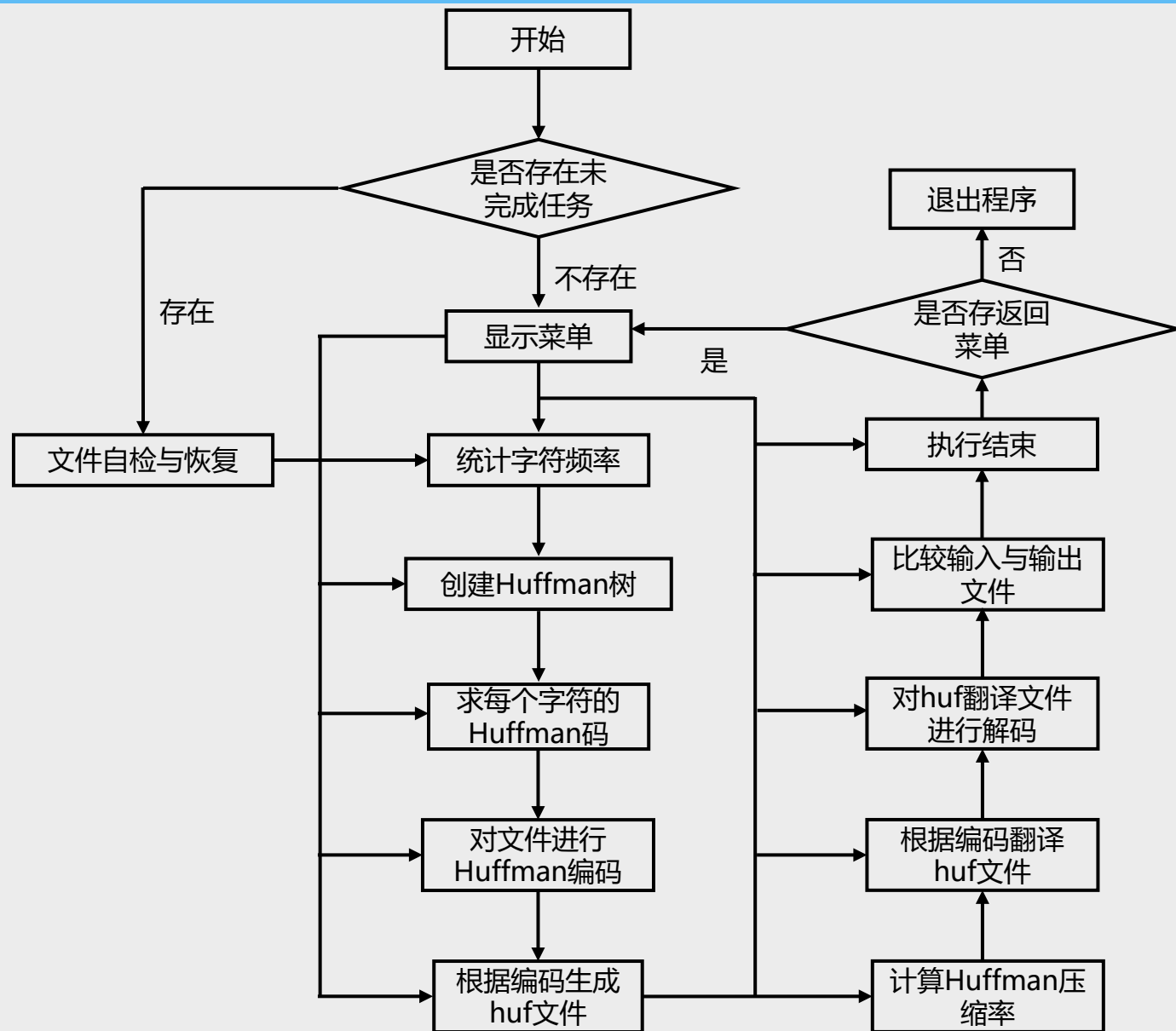
3 程序执行逻辑

4 函数功能



PART 2

1. 显示菜单;
2. 统计字符频率;
3. 创建Huffman树;
4. 求每个字符的Huffman码;
5. 对文件进行Huffman编码;
6. 根据编码生成huf文件;
7. 计算Huffman压缩率;
8. 根据编码翻译huf文件;
9. 对huf翻译文件进行解码;
10. 比较输入文件与输出文件;
11. 文件自检与恢复;
12. 读入未完成文件;



- (1) 程序启动，首先检测是否有上一次未完成任务
如果有，则询问是否继续，跳转至第（4）步开始生成Huffman树；如果没有，
则跳转至第（2）步显示开始菜单。
- (2) 显示菜单，询问要执行哪一步操作。当输入相应数字后，程序直接跳转。
- (3) 统计字符频率。结束后询问是否继续下一步
如果选择是，则继续下一步；如果不是，则询问是否返回菜单
- (4) 创建Huffman树。
- (5) 求每个字符的Huffman码。
- (6) 对文件进行Huffman编码。
- (7) 根据编码生成huf文件。
- (8) 计算Huffman压缩率。
- (9) 根据编码翻译huf文件。
- (10) 对huf翻译文件进行解码。
- (11) 比较输入文件与输出文件。
- (12) 文件自检与恢复。
- (13) 读入未完成文件。
- (14) 程序执行结束，询问是否返回菜单。
如果选择是，则返回菜单；如果不是，则退出程序

```
1. typedef struct {  
    int char_ASCII;  
    int value;  
    char *Huffmancode;  
    int Huffmancode_bit = 0;  
}ElemType;
```

函数说明：char_ASCII为字符所对应的ASCII值；value为字符的出现频率或权值；Huffmancode为Huffman编码所在数组的指针；Huffmancode bit为Huffmancode数组的长度。

功能：定义链表中的元素结构体

```
2. typedef struct node {  
    ElemType elem;  
    struct node *next;  
}LinkNode, *LinkList;
```

函数说明：链表结构体。elem为元素，*next为下一链表的地址。

功能：定义链表结构体。

使用方法：预定义，无需手动调用

3. Status List_Init(LinkList &L)

函数说明：链表的初始化操作

功能：创建一个可用的链表并将其初始化

使用方法：输入LinkList类型的链表L

4. Status List_Insert(LinkList &L, int i, ElemType e)

函数说明：链表的插入操作

功能：将一个元素插入到链表的指定位置

使用方法：输入LinkList类型的链表L，插入位置i，要插入的ElemType类型的元素e

5. Status List_Destroy(LinkList &L)

函数说明：删除链表，将整个链表空间摧毁

功能：删除给定的链表L

使用方法：输入一个LinkList类型的L变量，摧毁L

6. typedef struct {

unsigned int value, ASCII_CODE;

unsigned int parent, lchild, rchild;

}HTNode, *HuffmanTree;

函数说明：Huffman树结构体定义，value为叶子结点的权值；

ASCII_CODE为叶子结点所对应的字符的ASCII码；parent、lchild、rchild为双亲、左孩子、右孩子

功能：定义Huffman树的结构体

使用方法：由其他函数调用，无需手动调用

7. Status File_sourceload()

功能：读取源文件信息，统计字符出现频率

8. Status File_read_char_num(LinkList &L, int &text_length, int &char_number)

功能：读取huffman_temp\\char_frequency.txt中的信息

9. Status HuffmanTree_Select(HuffmanTree HT, int n, int &s1, int &s2)

功能：选择权值最小的两个结点

10. Status HuffmanTree_Create(HuffmanTree &HT, int n, LinkList L, int if_print)

功能：创建Huffman树

11. Status HuffmanTree_Code(HuffmanTree HT, int n, LinkList &L)

功能：从叶子到根逆向求每个字符的Huffman编码，储存在指针L.Huffmancode中

12. Status HuffmanCode_Write(HuffmanTree HT, int n, LinkList &L)

功能：存储Huffman编码为字典

13. `char * HuffmanCode_CharPoint(LinkList L, char c)`

功能：返回某个字符的Huffman编码所在的指针

14. `Status HuffmanCode_Encode(HuffmanTree HT, LinkList L, int if_print)`

功能：对文件进行Huffman编码，存储在huffman_encode.txt中

15. `char bit_number_transform(int n[])`

功能：每8位转化为对应的ASCII码

16. `Status text_huffmancode_create()`

功能：生成huf文件

17. `Status text_huffmancode_unzip()`

功能：解压huf文件

18. `Status HuffmanCode_FileDecode(HuffmanTree HT, int char_number, int text_length)`

功能：对文件进行Huffman解码，存储在text_decode.txt中

19. Status File_Error_Percentage()

功能：对输入文件和输出文件进行比较，求正确率

20. Status File_Zip_Percentage()

功能：计算压缩率

21. void menu()

功能：显示菜单

22. int if_go_next()

功能：询问是否进行下一步操作

23. void if_backto_menu()

功能：询问是否返回菜单

24. void menu_function1()

功能：菜单功能1，按流程执行所有操作

25. void menu_function2()

功能：菜单功能2，统计字符频率

26. void menu_function3()

功能：菜单功能3，创建Huffman树

27. void menu_function4()

功能：菜单功能4，求每个字符的Huffman编码

28. void menu_function5()

功能：菜单功能5，对文件进行Huffman编码

29. void menu_function6()

功能：菜单功能6，根据编码生成.huf文件

30. void menu_function7()

功能：菜单功能7，计算Huffman压缩率

31. void menu_function8()

功能：菜单功能8，根据编码翻译.huf文件

32. void menu_function9()

功能：菜单功能9，对翻译得到的文件进行解码

33. void menu_function10()

功能：菜单功能10，比较输入文件与输出文件

34. void menu_function11()

功能：菜单功能11，退出程序

35. void Project_Recovery()

功能：任务恢复。重新读入未完成的char_frequency.txt文档，继续执行任务。

36. void main()

函数说明：main函数

功能：判断程序目录下是否存在未完成的工程，若有则继续执行，若没有则跳转至菜单。

使用方法：程序入口，无需手动执行。

实验结果与分析

1 程序执行结果

2 结果分析

3 文件压缩率

4 译码准确率

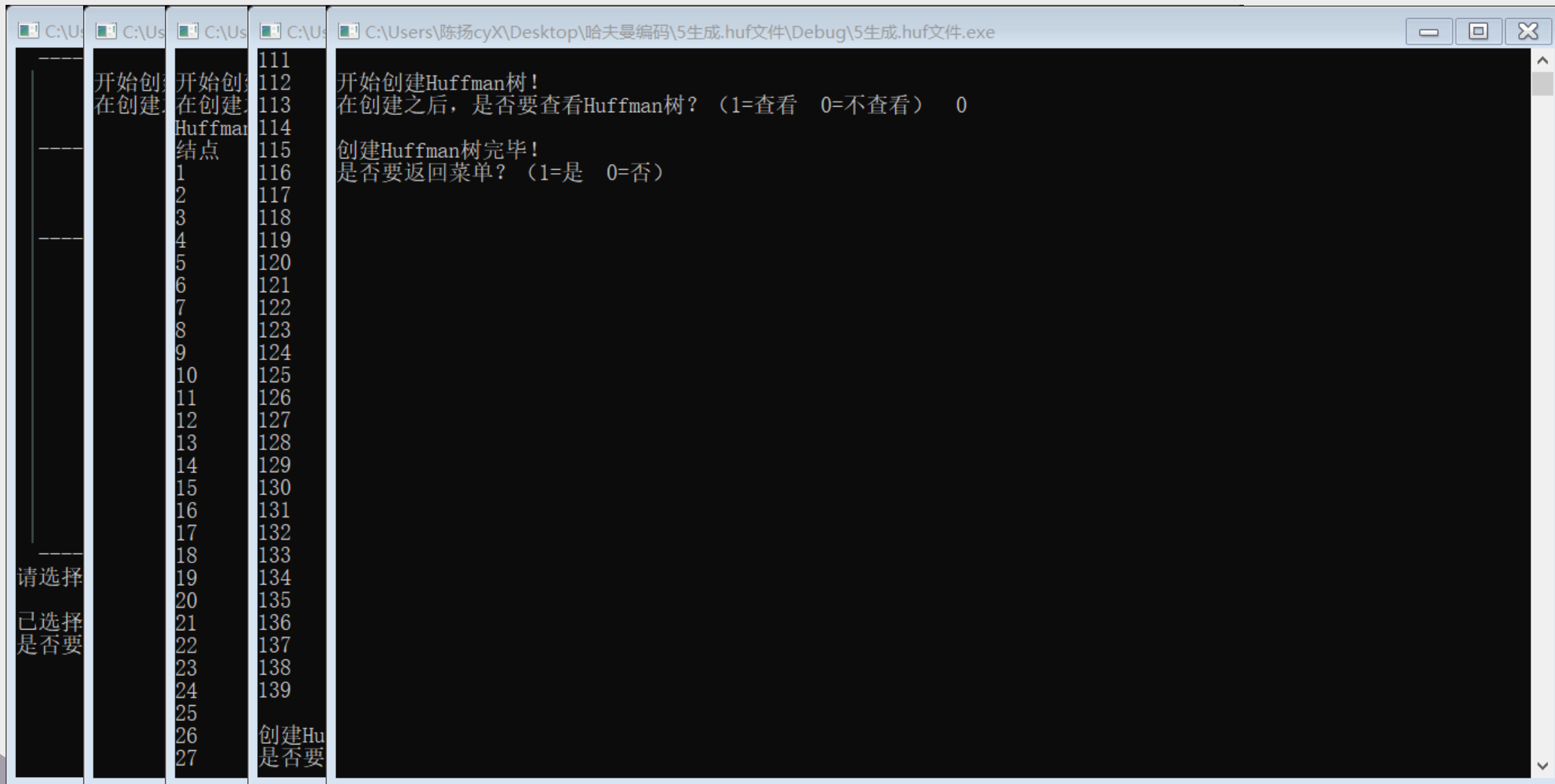


PART 3

```
C:\Users\陈扬cyX\Desktop\哈夫曼编码\5生成.huf文件\Debug\5生成.huf文件.exe
开始统计字符频率！
文本字符统计完毕！
信息保存在C:\Users\陈扬cyX\Desktop\哈夫曼编码\5生成.huf文件\5生成.huf文件\huffman_temp\char_frequency.txt

文本长度：8681    字符种类：70
按照ASCII码排列，字符出现频率如下：
换行： 56
空格：1372
": 56      ' : 17      ( : 5
): 5       , : 67      - : 13      . : 58      0 : 9
1 : 4      2 : 5      3 : 3      4 : 2      5 : 4
6 : 1      7 : 2      8 : 1      9 : 1      : : 4
? : 2      A : 16     B : 11     C : 7      D : 14
E : 4      F : 1      G : 3      H : 14     I : 15
J : 6      K : 16     L : 2      M : 38     N : 12
O : 2      P : 6      R : 5      S : 24     T : 31
U : 13     W : 4      Y : 1      [ : 1      ] : 1
a : 566    b : 92      c : 221    d : 279    e : 825
f : 121    g : 126     h : 296    i : 534    j : 7
k : 26     l : 294     m : 175    n : 499    o : 555
p : 129    q : 8       r : 437    s : 481    t : 618
u : 165    v : 67     w : 105    x : 13     y : 108

请选择需 请选择需
已选择操 统计字符完毕！
是否要进 是否要返回菜单？（1=是 0=否）
```

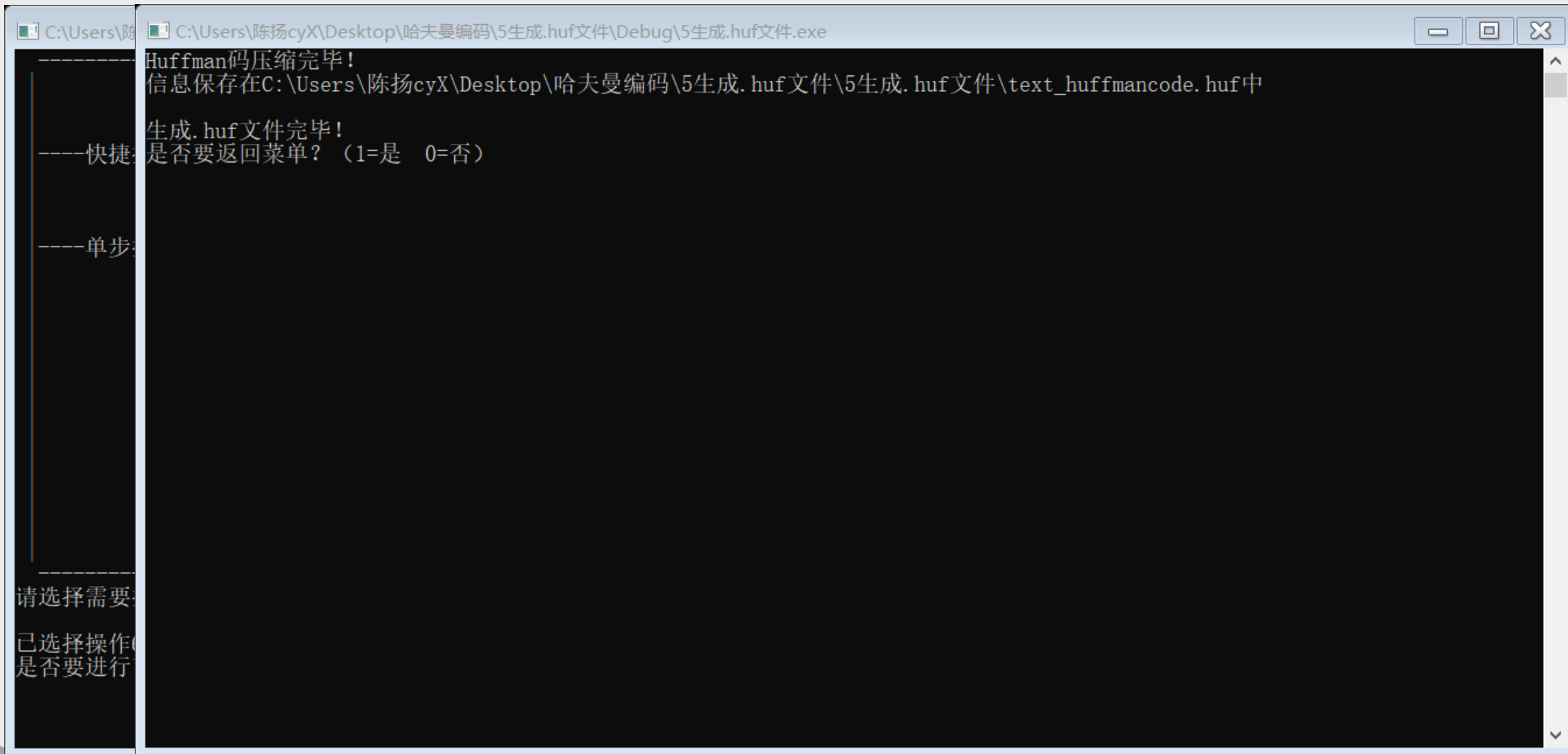


```
C:\Users\陈扬cyX\Desktop\哈夫曼编码\5生成.huf文件\Debug\5生成.huf文件.exe
111
112 开始创建Huffman树!
113 在创建之后, 是否要查看Huffman树? (1=查看 0=不查看) 0
114
115 创建Huffman树完毕!
116 是否要返回菜单? (1=是 0=否)
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
280
281
282
283
284
285
286
287
288
289
290
291
292
293
294
295
296
297
298
299
300
301
302
303
304
305
306
307
308
309
310
311
312
313
314
315
316
317
318
319
320
321
322
323
324
325
326
327
328
329
330
331
332
333
334
335
336
337
338
339
340
341
342
343
344
345
346
347
348
349
350
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
432
433
434
435
436
437
438
439
440
441
442
443
444
445
446
447
448
449
450
451
452
453
454
455
456
457
458
459
460
461
462
463
464
465
466
467
468
469
470
471
472
473
474
475
476
477
478
479
480
481
482
483
484
485
486
487
488
489
490
491
492
493
494
495
496
497
498
499
500
501
502
503
504
505
506
507
508
509
510
511
512
513
514
515
516
517
518
519
520
521
522
523
524
525
526
527
528
529
530
531
532
533
534
535
536
537
538
539
540
541
542
543
544
545
546
547
548
549
550
551
552
553
554
555
556
557
558
559
560
561
562
563
564
565
566
567
568
569
570
571
572
573
574
575
576
577
578
579
580
581
582
583
584
585
586
587
588
589
590
591
592
593
594
595
596
597
598
599
600
601
602
603
604
605
606
607
608
609
610
611
612
613
614
615
616
617
618
619
620
621
622
623
624
625
626
627
628
629
630
631
632
633
634
635
636
637
638
639
640
641
642
643
644
645
646
647
648
649
650
651
652
653
654
655
656
657
658
659
660
661
662
663
664
665
666
667
668
669
670
671
672
673
674
675
676
677
678
679
680
681
682
683
684
685
686
687
688
689
690
691
692
693
694
695
696
697
698
699
700
701
702
703
704
705
706
707
708
709
710
711
712
713
714
715
716
717
718
719
720
721
722
723
724
725
726
727
728
729
730
731
732
733
734
735
736
737
738
739
740
741
742
743
744
745
746
747
748
749
750
751
752
753
754
755
756
757
758
759
760
761
762
763
764
765
766
767
768
769
770
771
772
773
774
775
776
777
778
779
780
781
782
783
784
785
786
787
788
789
790
791
792
793
794
795
796
797
798
799
800
801
802
803
804
805
806
807
808
809
810
811
812
813
814
815
816
817
818
819
820
821
822
823
824
825
826
827
828
829
830
831
832
833
834
835
836
837
838
839
840
841
842
843
844
845
846
847
848
849
850
851
852
853
854
855
856
857
858
859
860
861
862
863
864
865
866
867
868
869
870
871
872
873
874
875
876
877
878
879
880
881
882
883
884
885
886
887
888
889
890
891
892
893
894
895
896
897
898
899
900
901
902
903
904
905
906
907
908
909
910
911
912
913
914
915
916
917
918
919
920
921
922
923
924
925
926
927
928
929
930
931
932
933
934
935
936
937
938
939
940
941
942
943
944
945
946
947
948
949
950
951
952
953
954
955
956
957
958
959
960
961
962
963
964
965
966
967
968
969
970
971
972
973
974
975
976
977
978
979
980
981
982
983
984
985
986
987
988
989
990
991
992
993
994
995
996
997
998
999
1000
```

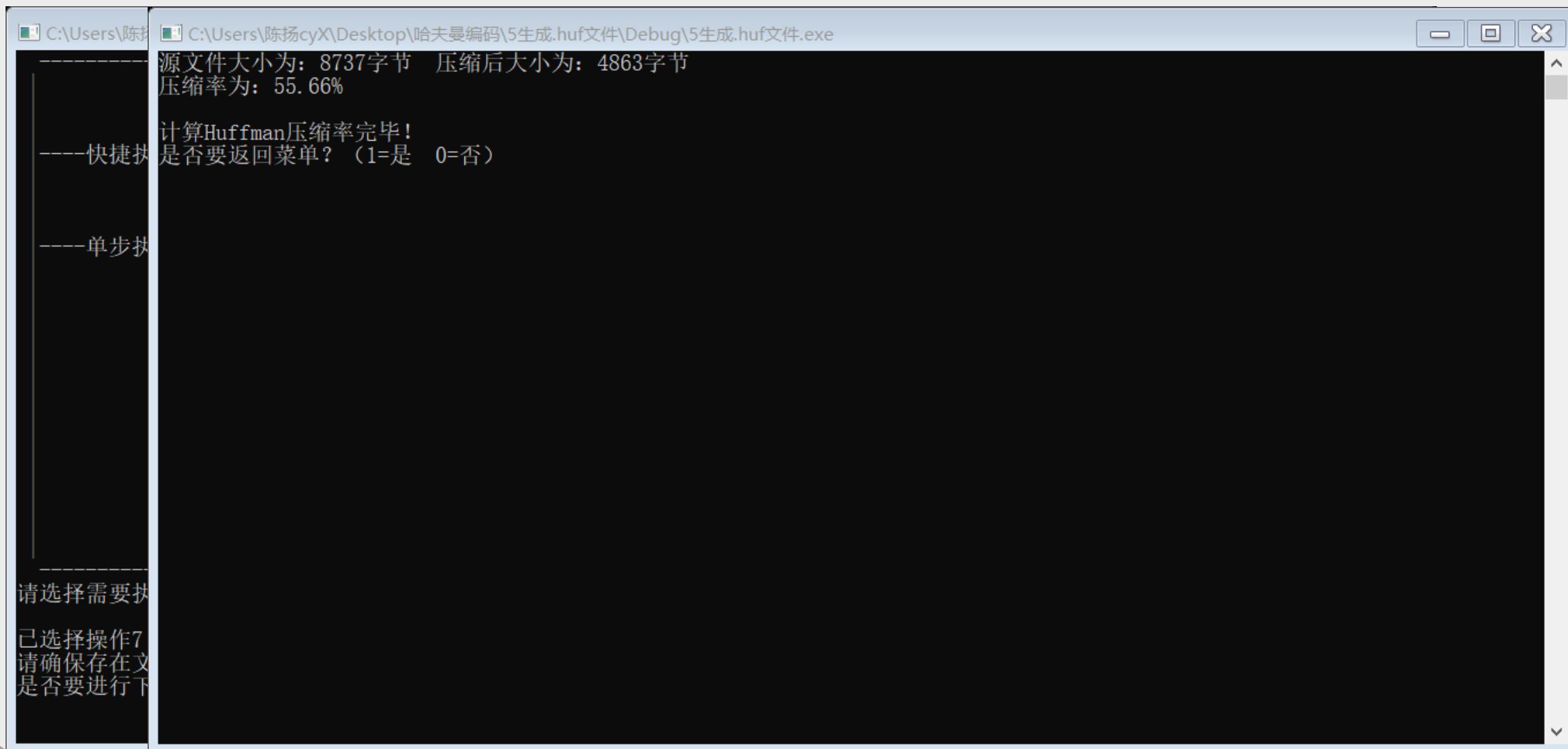
```
C:\Users\陈扬cyX\Desktop\哈夫曼编码\5生成.huf文件\Debug\5生成.huf文件.exe
-----
Huf  Huffman编码完毕!
----快捷执行---- 信息保存在C:\Users\陈扬cyX\Desktop\哈夫曼编码\5生成.huf文件\5生成.huf文件\huffman_temp\huffman_codeinfo.txt
                    各字符对应的Huffman编码为:
                    换行: 0011010
                    空格: 111
                    ",: 0011011
                    ',: 100001101
                    (: 10000111100
                    ): 10000111101
                    ,: 1000001
                    -: 1101110100
                    .: 0101000
                    0: 1000011101
                    1: 10000001110
                    2: 10000111110
                    3: 110110010101
                    4: 1101110001111
                    5: 10000001111
                    6: 1000011100100
                    7: 100001100110
                    8: 1000011100101
                    9: 1000011100110
                    :: 10000110000
----单步执行---- ? : 100001100111
                    A: 100000001
                    B: 1101100100
                    C: 0101001010
                    D: 1101110111
                    E: 10000110001
                    F: 1000011100111
                    G: 1101110001110
                    2 H: 010100100
                    I: 100000000
                    J: 11011001011
                    K: 100000010
                    3 L: 100001110000
                    M: 11011000
                    N: 1101110000
                    O: 100001110001
                    4 P: 11011100010
                    R: 10000111111
                    S: 110110011
                    T: 01010011
                    5 U: 1101110101
                    W: 10000110010
                    Y: 1101100101000
                    [: 1101100101001
                    6 ]: 1101110001110
                    a: 1010
                    b: 1101101
                    c: 00111
                    7 d: 10001
                    e: 000
                    f: 010101
                    g: 010110
                    8 h: 10111
                    i: 0111
                    j: 0101001011
                    k: 110111001
                    9 l: 10110
                    m: 110101
                    n: 0110
                    o: 1001
                    10 p: 010111
                    q: 1000000110
                    r: 0010
                    s: 0100
                    t: 1100
                    u: 110100
                    v: 1000010
                    w: 1101111
                    11 x: 1101110110
                    y: 001100
                    文件写入完毕!

请选择需要执行的操作: 1=求每个字符的Huffman编码完毕! 0=是否要返回菜单? (1=是 0=否)
已选择操作4! 请确认是否要进行下一步操作:
```

```
C:\Users\陈扬cyX\Desktop\哈夫曼编码\5生成.huf文件\Debug\5生成.huf文件.exe
在创建文件
开始
在创建文件
1000 Huffman编码完毕!
1110 信息保存在C:\Users\陈扬cyX\Desktop\哈夫曼编码\5生成.huf文件\5生成.huf文件\huffman_temp\huffman_encode.txt
1000 对文件进行Huffman编码完毕!
1110 是否要返回菜单? (1=是 0=否)
1000 0
0111 01011
1100 10011
0010 01001
0011 10101
1011 00110
0100 11010
0010 01000
0111 00100
1111 10001
1100 11011
1100 10011
1011 11011
0011 01001
0010 10111
1011 00001
0011 11100
0111 00000
0100 11100
0000 01100
0011 Huffman
1000 信息保
1011 文件H
0101
```



```
C:\Users\陈扬cyX\Desktop\哈夫曼编码\5生成.huf文件\Debug\5生成.huf文件.exe
Huffman码压缩完毕!
信息保存在C:\Users\陈扬cyX\Desktop\哈夫曼编码\5生成.huf文件\5生成.huf文件\text_huffmancode.huf中
生成.huf文件完毕!
----快捷: 是否要返回菜单? (1=是 0=否)
----单步:
请选择需要:
已选择操作0
是否要进行
```

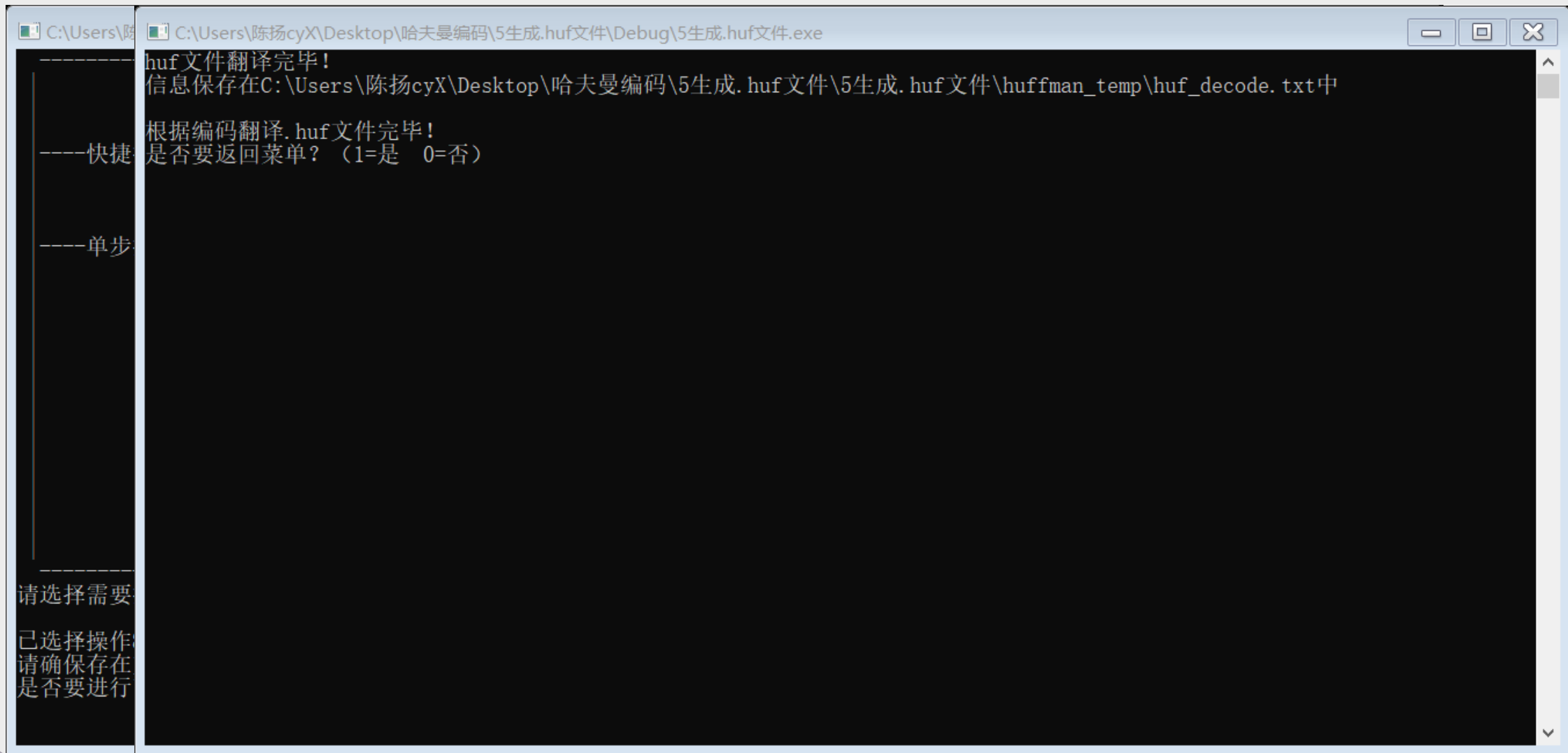


```
C:\Users\陈扬cyX\Desktop\哈夫曼编码\5生成.huf文件\Debug\5生成.huf文件.exe
-----
源文件大小为: 8737字节  压缩后大小为: 4863字节
压缩率为: 55.66%

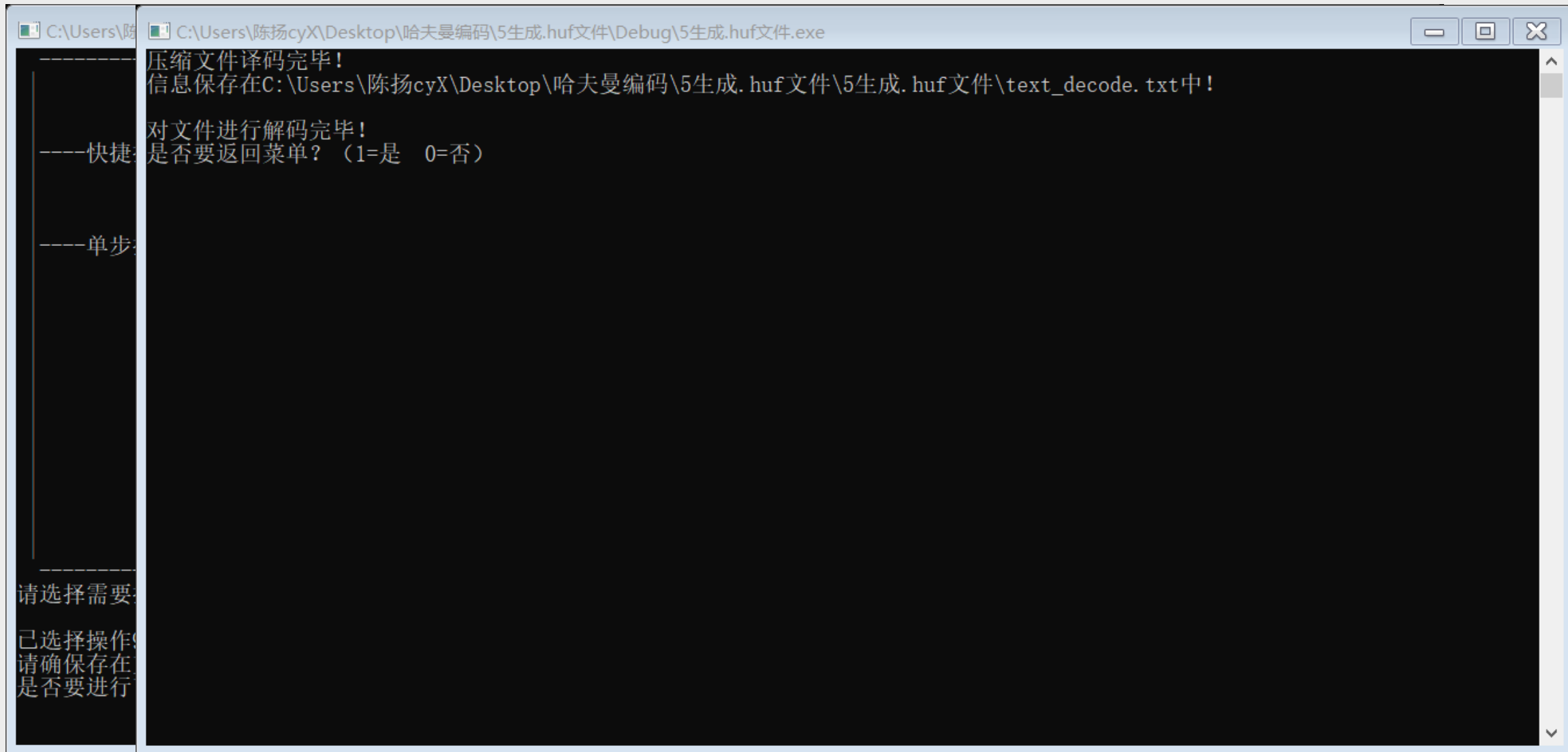
计算Huffman压缩率完毕!
----快捷操作  是否要返回菜单? (1=是  0=否)

----单步执行

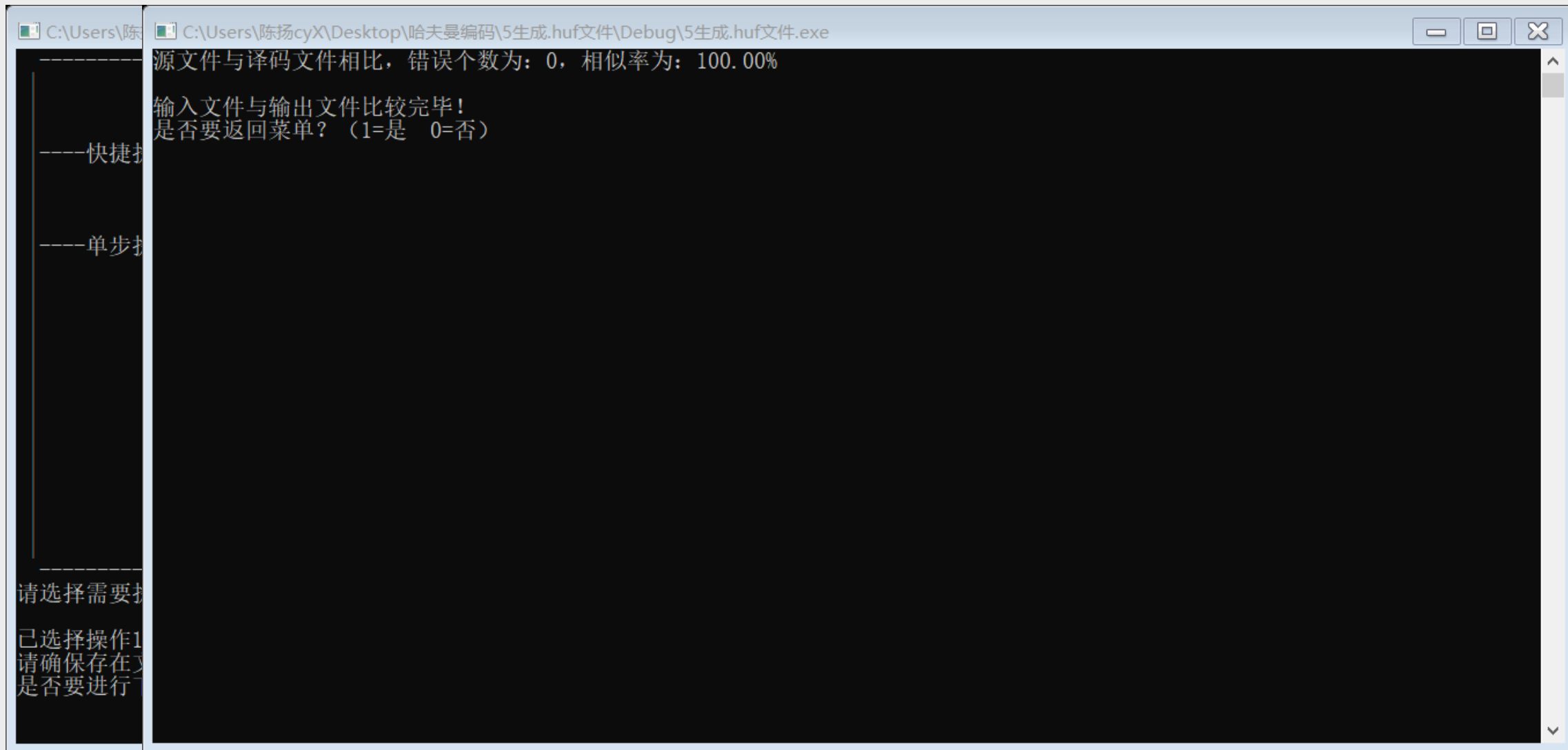
-----
请选择需要执行的操作:
已选择操作7
请确保存在文件
是否要进行下一步?
```

```
C:\Users\陈扬cyX\Desktop\哈夫曼编码\5生成.huf文件\Debug\5生成.huf文件.exe
huf文件翻译完毕!
信息保存在C:\Users\陈扬cyX\Desktop\哈夫曼编码\5生成.huf文件\5生成.huf文件\huffman_temp\huf_decode.txt中
根据编码翻译.huf文件完毕!
----快捷 是否要返回菜单? (1=是 0=否)
----单步
请选择需要
```



```
C:\Users\陈扬cyX\Desktop\哈夫曼编码\5生成.huf文件\Debug\5生成.huf文件.exe
压缩文件译码完毕!
信息保存在C:\Users\陈扬cyX\Desktop\哈夫曼编码\5生成.huf文件\5生成.huf文件\text_decode.txt中!
对文件进行解码完毕!
----快捷: 是否要返回菜单? (1=是 0=否)
----单步:
-----
请选择需要:
已选择操作:
请确保存在:
是否要进行:
```



```
C:\Users\陈扬cyX\Desktop\哈夫曼编码\5生成.huf文件\Debug\5生成.huf文件.exe
源文件与译码文件相比，错误个数为：0，相似率为：100.00%
输入文件与输出文件比较完毕！
是否要返回菜单？（1=是 0=否）

快捷
单步

请选择需要
已选择操作1
请确保存在
是否要进行
```

11.按流程执行所有操作

选择该操作，即按照顺序执行菜单中的“2.统计字符频率”、“3.创建Huffman树”、“4.求每个字符的Huffman编码”、“5.对文件进行Huffman编码”、“6.根据编码生成.huf文件”、“7.计算Huffman压缩率”、“8.根据编码翻译.huf文件”、“9.对翻译得到的文件进行解码”、“10.比较输入文件与输出文件”

12.退出程序

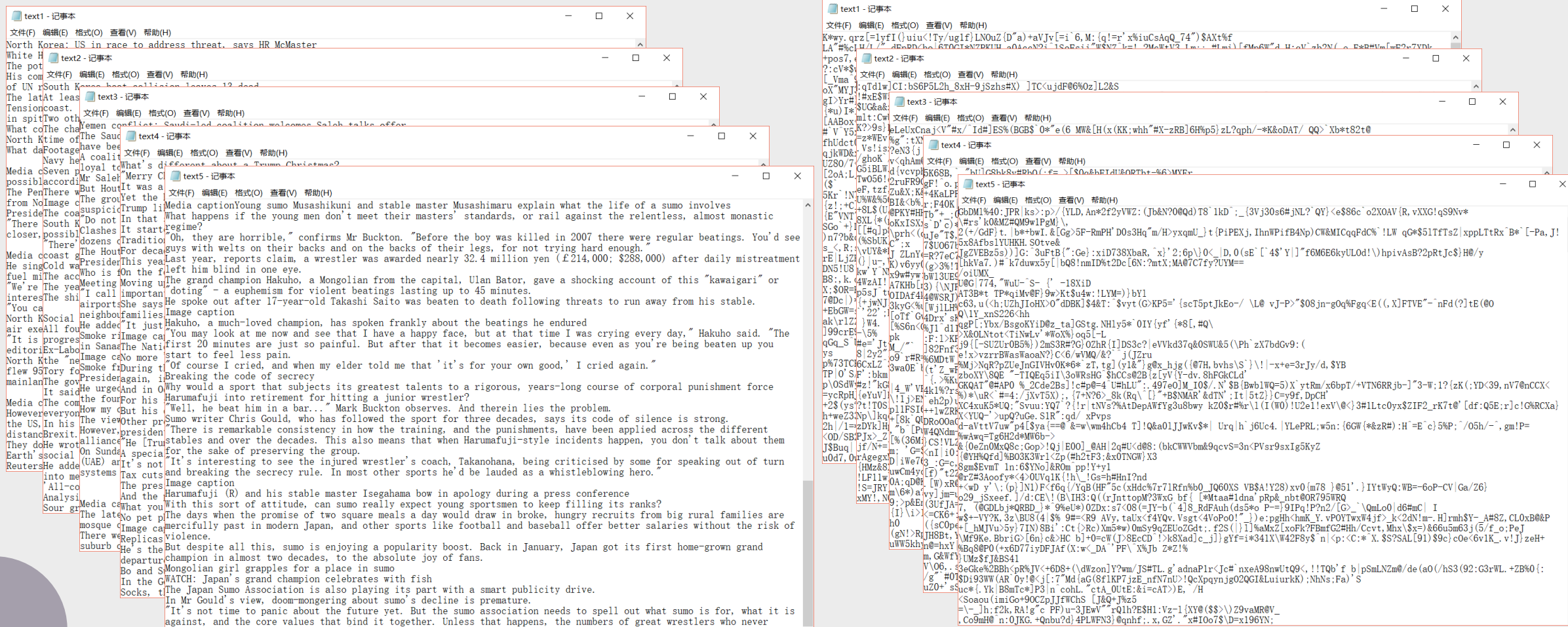
选择该操作，即执行exit()函数，完全关闭并退出程序。

为确保程序文件的不丢失及恢复，引入了“对未完成任务的恢复处理”操作，该操作会在程序启动时自动执行。

通过判断程序目录下huffman_temp\char_number.txt文件是否存在，来判断上次运行。若该文件存在，则说明程序已经运行过，且留有信息文档。此时，程序将会询问用户，是否继续上一次的任务。如果输入1，确定继续执行，则程序将会跳过菜单步骤，直接读取huffman_temp\char_number.txt的信息，按顺序执行“3.创建Huffman树”、“4.求每个字符的Huffman编码”、“5.对文件进行Huffman编码”、“6.根据编码生成huf文件”、“7.计算Huffman压缩率”、“8.根据编码翻译huf文件”、“9.对翻译得到的文件进行解”、“10.比较输入文件与输出文件”。

```
已经重新加载该任务！
是否要进行下一步操作？（1=是 0=否）
```

使用Huffman编码对文本进行压缩的效果与多方面因素有关。**文本长度、字符种类及字符的出现频率（权值）**对文件的压缩率有着很大的影响。



规则文本

序号	文本长度	字符种类	源文件大小	huf文件大小	压缩率	错误个数	准确率
1	4196	69	4228字节	2380字节	56.29%	0	100.00%
2	6299	64	6354字节	3537字节	55.67%	0	100.00%
3	8424	67	8496字节	4770字节	56.14%	0	100.00%
4	3989	67	4032字节	2268字节	56.25%	0	100.00%
5	9448	72	9517字节	5361字节	56.33%	0	100.00%
6	6432	80	6535字节	3803字节	58.37%	0	100.00%
7	380955	94	381465字节	263758字节	69.14%	0	100.00%
8	74351	93	76163字节	45393字节	59.60%	0	100.00%
9	161368	96	168965字节	102226字节	60.50%	0	100.00%
10	61574	88	62422字节	39472字节	63.23%	0	100.00%

不规则文本

序号	文本长度	字符种类	源文件大小	huf文件大小	压缩率	错误个数	准确率
1	6000	95	6063字节	4955字节	81.73%	0	100%
2	12000	95	12113字节	9934字节	82.01%	0	100%
3	20000	95	20230字节	16557字节	81.84%	0	100%
4	25000	95	25236字节	20726字节	82.13%	0	100%
5	30000	95	30341字节	24869字节	81.96%	0	100%
6	35000	95	35355字节	29028字节	82.10%	0	100%
7	40000	95	40434字节	33166字节	82.03%	0	100%
8	100000	95	101025字节	83016字节	82.17%	0	100%
9	500000	95	505214字节	415494字节	82.24%	0	100%
10	1000000	95	1010493字节	831125字节	82.25%	0	100%
11	2000000	95	2021031字节	1662545字节	82.26%	0	100%
12	3000000	95	3031503字节	2493853字节	82.26%	0	100%
13	4000000	95	4042218字节	3325383字节	82.27%	0	100%
14	5000000	95	5052347字节	4156946字节	82.28%	0	100%
15	10000000	95	10105511字节	8314397字节	82.28%	0	100%

通过以上25篇不同文档的测试，发现本程序在Huffman编码/译码方面有着较好的性能，基本能够实现100%的译码准确率。

但是，这并不代表本程序对于所有的文档译码都能够准确。在进行本程序测试时，未进行如下测试：

- (1) 对**中文文档**的编码/译码

- (2) 对ASCII码值不在31~126之间的字符的编码/译码

因此，本程序只能够基本保证对可显示字符（ASCII码在31~126之间的字符）的编码/译码准确。

总结分析

1 程序设计评价

2 Huffman编码
/译码技术评价

3 自我评价



PART 4

程序能够实现题目中要求的全部功能，并且能够保证几乎为100%的译码准确率

但是，仍有几个缺点：

- 1.由于系统特性，中文的编码方式与字符不同，无法使用Huffman算法进行编码，译码。
- 2.因时间与精力有限，没有对所有的ASCII字符进行测试。
- 3.因设备限制，没有将程序在不同的设备上运行，难以保证程序的兼容性、稳定性、准确性。
- 4.没有为程序编写GUI界面，非作者的用户使用起来仍需要一定时间熟悉程序的执行流程。
- 5.对于程序的执行逻辑、错误应对办法没有很好的优化，仍然存在bug。

- 1.Huffman编码/译码技术对于电文的传输、压缩有着一定的帮助作用，能够在一定程度上缩小电文的占用空间。
- 2.Huffman编码技术对于字符规则的文档有较低的压缩效率，能够压缩至原文件大小的50%~60%，是一项很不错的压缩技术。
- 3.Huffman编码技术对于字符不规则的文档的压缩效果并不明显，压缩率为82%左右，相比其他压缩技术，效果并不突出。
- 4.利用Huffman编码技术压缩文档时，需要占用大量的内存空间，可能引起程序的停止运行，仍需进一步优化，找出解决方案。

- 1.能够更加熟练的掌握C语言的文件读写操作
- 2.对报告的撰写技巧的理解与完善
- 3.能够独立的思考并解决程序编写过程中的问题
- 4.对Huffman树、Huffman编码的理解加深
- 5.数据结构有很广泛以及很重要的应用

The background features a stylized tree with a blue trunk and branches, adorned with green, grey, and yellow hexagonal leaves. Large, colorful circles in shades of teal, orange, and light green are scattered across the slide.

谢谢!

信息院 网络工程161班
陈扬 19316117