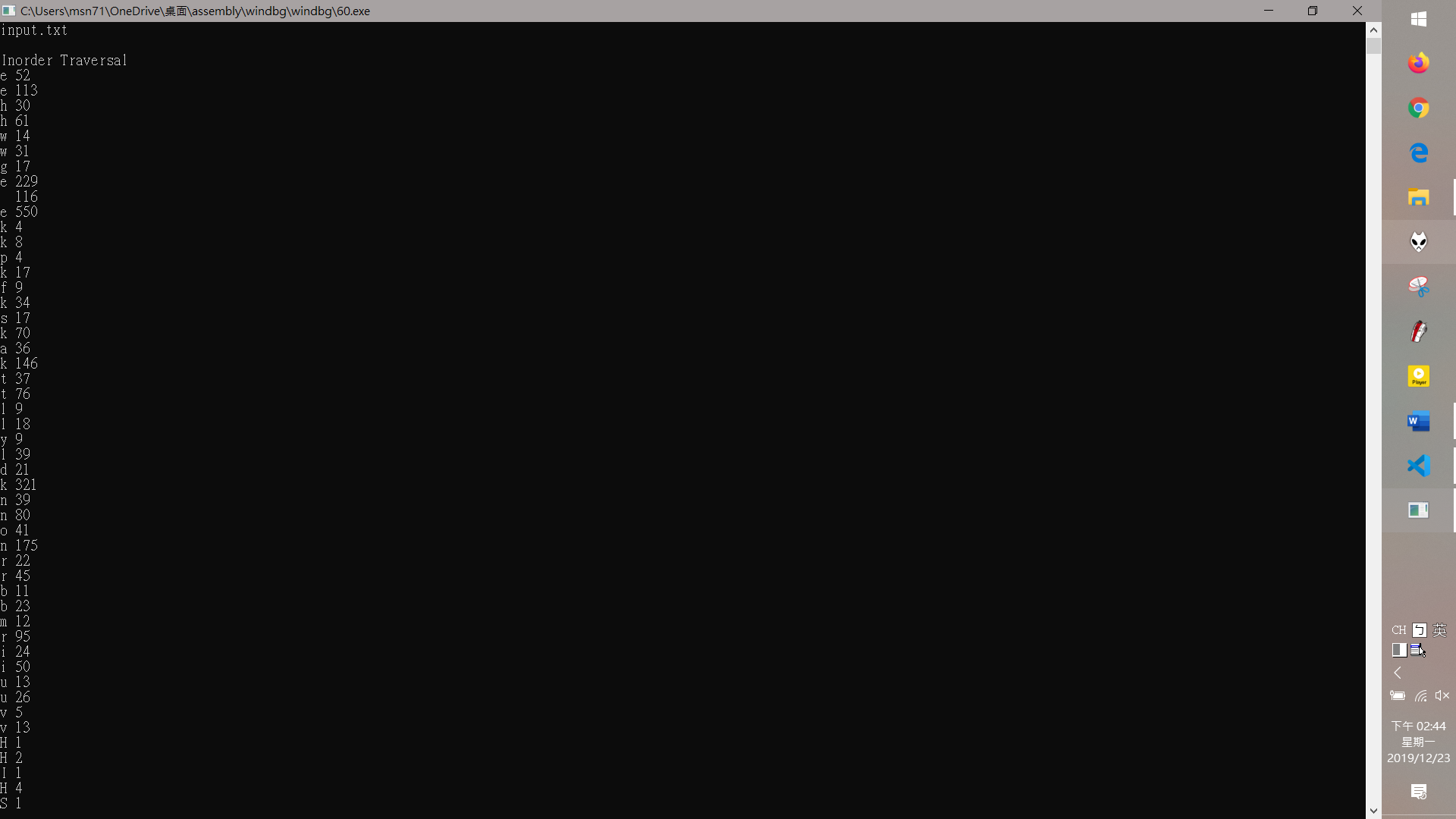
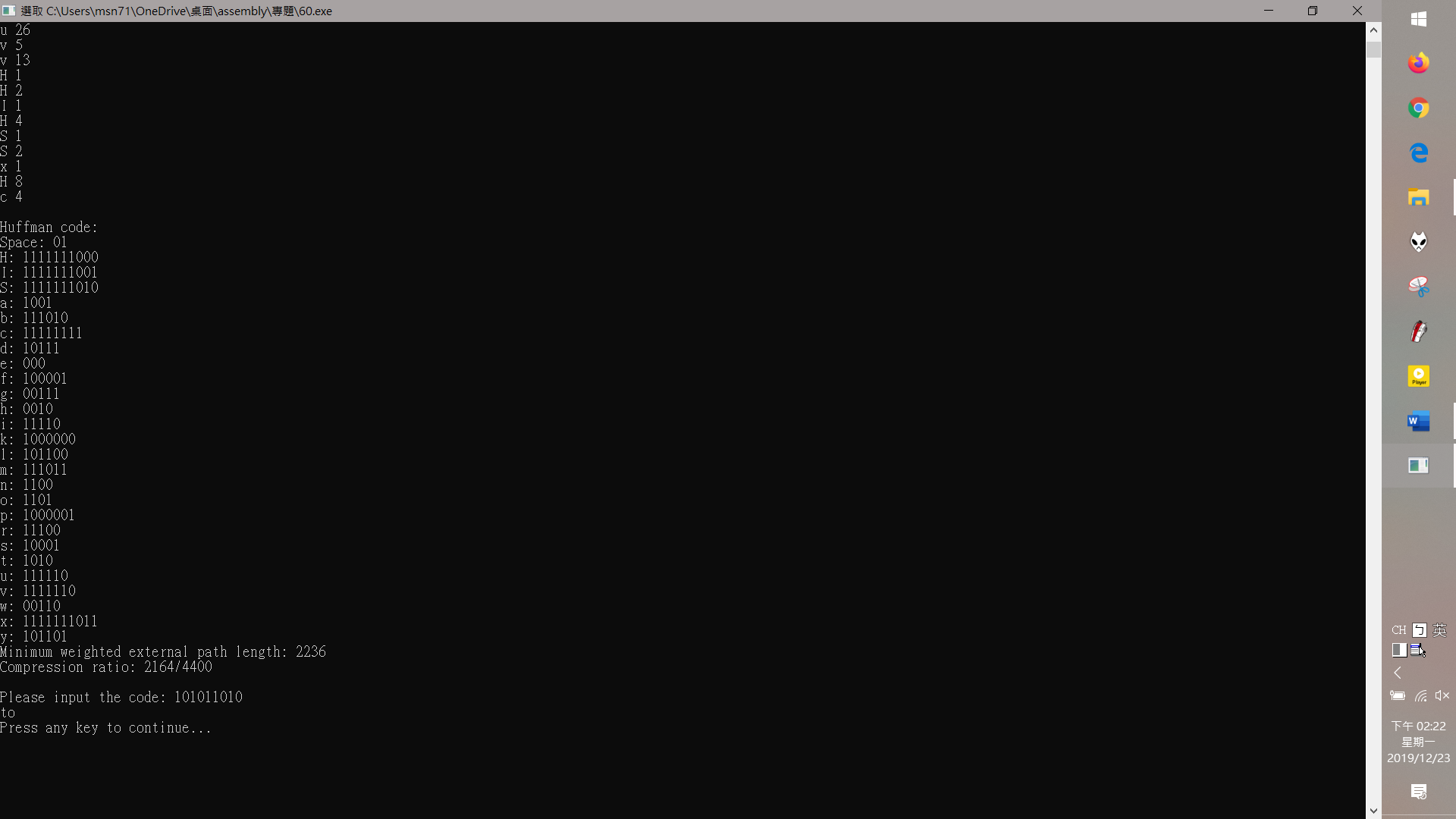
* + - 1. 報告標題:lab12
      2. 組別、姓名:no.60 呂旻翰 郭忠明
      3. 程式執行流程、記憶體（暫存器）狀態截圖





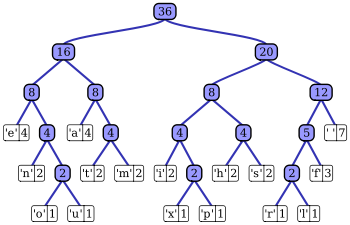
* + - 1. 截圖說明、程式碼說明 (700行 很長 裡面有註解 但會在說明各個function怎麼實作的)

1. TITLE Lab of week2              (week2.ASM)
2. ; This program locates the cursor and displays the
3. ; system time. It uses two Win32 API structures.
4. ; Last update: 6/30/2005
5. INCLUDE Irvine32.inc
6. INCLUDE Macros.inc
7. Max = 5000
8. ; Redefine external symbols for convenience
9. ; Redifinition is necessary for using stdcall in .model directive
10. ; using "start" is because for linking to WinDbg.  added by Huang
12. main          EQU start@0
13. ;Comment @
14. ;Definitions copied from SmallWin.inc:
15. GetProcessHeap proto
16. HeapAlloc proto,hHeap:HANDLE,dwFlags:DWORD,dwBytes:DWORD
17. sort proto,f:dword
18. tprint proto,r:dword
19. searchleaf proto,rr:dword,tt:dword,vv:dword,cc:dword
20. tablesort proto,ff:dword
21. print proto,p:dword
22. htree struct
23. lab dword 0
24. fre dword 0
25. left dword 0
26. right dword 0
27. slink dword 0
28. htree ends
29. table struct
30. key dword 0
31. lab dword 0
32. count dword 0
33. fre dword 0
34. link dword 0
35. table ends
37. .data
39. first dword 0
40. hHeap HANDLE ?
41. btemp dword 0
42. temp dword 0
43. root dword 0
44. map dword 0 ;code like 01010101
45. output1 byte "Huffman code:",0
46. pmap dword 0 ;ptr map = table
47. sum dword 0
48. cfre dword 0 ;count frequency
49. output2 byte "Minimum weighted external path length: ",0
50. output3 byte "Compression ratio: ",0
51. output4 byte "Please input the code: ",0
52. cnum dword 0
53. inputc2 byte Max+1 dup (?)
54. output5 byte "Space: ",0
55. output6 byte "Inorder Traversal",0
56. inputc byte Max+1 dup (?)
57. filename byte Max+1 dup (?)
58. fileHandle HANDLE ?
59. .code
60. main PROC
61. mov edx,offset filename
62. mov ecx,Max
63. call ReadString
64. mov edx,offset filename
65. mov ecx,sizeof filename
66. call OpenInputFile
67. mov fileHandle,eax
68. mov edx,offset inputc
69. mov ecx,max
70. call ReadFromFile
71. mov esi,offset inputc
72. .while byte ptr [esi]!=0
73. ;排除除了英文跟空格的字
74. .if byte ptr [esi]==' ' || (byte ptr [esi]>='A' && byte ptr [esi]<='Z') || (byte ptr [esi]>='a' && byte ptr [esi]<='z')
75. ;創建初始連結使其成為struct
76. .if first==0
77. invoke GetProcessHeap
78. mov hHeap,eax
79. invoke HeapAlloc,hHeap,HEAP\_ZERO\_MEMORY,type htree  ;first=(ptree)malloc(sizeof(\*first));
80. mov first,eax
81. mov eax,0
82. mov edi,first
83. mov (htree ptr [edi]).slink,eax                 ;first->slink=NULL;
84. mov (htree ptr [edi]).left,eax                  ;first->left=NULL;
85. mov (htree ptr [edi]).right,eax                 ;first->right=NULL;
86. mov eax,1
87. mov (htree ptr [edi]).fre,eax                       ;first->fre=1;
88. movzx eax,byte ptr [esi]
89. mov (htree ptr [edi]).lab,eax                       ;first->lab=(int)c;
90. ;
91. .else
92. mov eax,first                                       ;ptree btemp=first;
93. mov btemp,eax
95. .while btemp!=0
96. mov edi,btemp
97. mov al,byte ptr (htree ptr [edi]).lab
98. .if al== byte ptr [esi]
99. inc (htree ptr [edi]).fre                   ;btemp->fre++;
100. .break                                      ;break
101. .else
102. mov edi,btemp
103. mov eax,(htree ptr [edi]).slink         ;btemp=btemp->slink;
104. mov btemp,eax
105. .endif
106. .endw
107. .if btemp==0
108. mov eax,first                                   ;btemp=first;
109. mov btemp,eax
110. mov edi,btemp
111. mov eax,(htree ptr [edi]).slink
112. .while eax!=0                               ;btemp->slink!=NULL
113. mov edi,btemp
114. mov eax,(htree ptr [edi]).slink         ;btemp=btemp->slink;
115. mov btemp,eax
116. mov edi,btemp
117. mov eax,(htree ptr [edi]).slink
118. .endw
119. mov eax,0                                       ;ptree temp=NULL;
120. mov temp,eax
121. invoke GetProcessHeap
122. mov hHeap,eax                                   ;temp=(ptree)malloc(sizeof(\*temp));
123. invoke HeapAlloc,hHeap,HEAP\_ZERO\_MEMORY,type htree
124. mov temp,eax
125. mov edi,temp
126. mov eax,0
127. mov (htree ptr [edi]).slink,eax             ;temp->slink=NULL;
128. mov (htree ptr [edi]).left,eax                  ;temp->left=NULL;
129. mov (htree ptr [edi]).right,eax             ;temp->right=NULL;
130. mov eax,1
131. mov (htree ptr [edi]).fre,eax                   ;temp->fre=1;
132. movzx eax,byte ptr [esi]
133. mov (htree ptr [edi]).lab,eax                   ;temp->lab=(int)c;
134. mov eax,temp
135. mov edi,btemp
136. mov (htree ptr [edi]).slink,eax             ;btemp->slink=temp;
138. .endif
139. .endif
140. .endif
141. inc esi
142. .endw
144. ;mov eax,first  ;temp=first;
145. ;mov temp,eax
146. ;.while temp!=0
147. ;output
148. ;   mov edi,temp
149. ;   mov al,byte ptr (htree ptr [edi]).lab
150. ;   call WriteChar
151. ;   mov al,' '
152. ;   call WriteChar
153. ;   mov eax,(htree ptr [edi]).fre
154. ;   call WriteDec
155. ;   call Crlf
156. ;   mov eax,(htree ptr [edi]).slink         ;temp=temp->slink;
157. ;   mov temp,eax
158. ;.endw
160. invoke sort,addr first
161. ;call Crlf
162. ;mov eax,first  ;temp=first;
163. ;mov temp,eax
164. ;.while temp!=0
165. ;output
166. ;   mov edi,temp
167. ;   mov al,byte ptr (htree ptr [edi]).lab
168. ;   call WriteChar
169. ;   mov al,' '
170. ;   call WriteChar
171. ;   mov eax,(htree ptr [edi]).fre
172. ;   call WriteDec
173. ;   call Crlf
174. ;   mov eax,(htree ptr [edi]).slink         ;temp=temp->slink;
175. ;   mov temp,eax
176. ;.endw
177. ;建樹時同時進行排序 加入一次就排一次
178. mov eax,0
179. mov root,eax                                ;ptree root=NULL;
180. .while 1
181. invoke GetProcessHeap
182. mov hHeap,eax
183. invoke HeapAlloc,hHeap,HEAP\_ZERO\_MEMORY,type htree
184. mov temp,eax                            ;temp=(ptree)malloc(sizeof(\*temp));
186. mov eax,first
187. mov edi,temp
188. mov (htree ptr [edi]).left,eax          ;temp->left=first;
190. mov edi,first
191. mov eax,(htree ptr [edi]).slink
192. mov edi,temp
193. mov (htree ptr [edi]).right,eax         ;temp->right=first->slink;
195. mov edi,first
196. mov esi,(htree ptr [edi]).slink
197. mov eax,(htree ptr [esi]).fre
198. add eax,(htree ptr [edi]).fre
199. mov edi,temp
200. mov (htree ptr [edi]).fre,eax           ;temp->fre=first->slink->fre+first->fre;
202. mov edi,first
203. mov eax,(htree ptr [edi]).lab
204. mov edi,temp
205. mov (htree ptr [edi]).lab,eax           ;temp->lab=(int)first->lab;
207. mov edi,first
208. mov esi,(htree ptr [edi]).slink
209. mov eax,(htree ptr [esi]).slink
210. mov first,eax                           ;first=first->slink->slink;
212. .if first==0
213. mov eax,temp
214. mov root,eax                        ;root=temp;
215. .break                              ;break;
216. .endif
217. mov eax,first
218. mov edi,temp
219. mov (htree ptr [edi]).slink,eax         ;temp->slink=first;
221. mov eax,temp
222. mov first,eax                           ;first=temp;
224. invoke sort,addr first                  ;sort(&first);
226. .endw
227. call Crlf
228. mov edx,offset output6
229. call WriteString
230. call Crlf
231. invoke tprint,root
233. mov eax,0
234. mov map,eax                                 ;ptable map=NULL;
235. invoke searchleaf,root,addr map,0,0         ;searchleaf(root,&map,0,0);
237. ;output
238. ;call Crlf
239. ;mov eax,map    ;temp=map;
240. ;mov temp,eax
241. ;.while temp!=0
242. ;output
243. ;   mov edi,temp
244. ;   mov al,byte ptr (table ptr [edi]).lab
245. ;   call WriteChar
246. ;   mov al,' '
247. ;   call WriteChar
248. ;   mov eax,(table ptr [edi]).key
249. ;   call WriteDec
250. ;   call Crlf
251. ;   mov eax,(table ptr [edi]).link          ;temp=temp->link;
252. ;   mov temp,eax
253. ;.endw
254. ;output
256. invoke tablesort,addr map
258. call Crlf
259. mov edx,offset output1
260. call WriteString                            ;printf("Huffman code:\n");
261. call Crlf
263. invoke print,map
265. mov eax,map
266. mov pmap,eax                                ;ptable pmap=map;
267. mov eax,0
268. mov sum,eax
269. mov cfre,eax                                ;int sum=0,cfre=0;
270. ;算頻率總數去算壓縮率 與 總長
271. .while pmap!=0
272. mov eax,cfre
273. mov edi,pmap
274. add eax,(table ptr [edi]).fre
275. mov cfre,eax                            ;cfre=cfre+pmap->fre;
277. mov edi,pmap
278. mov eax,(table ptr [edi]).fre
279. mul (table ptr [edi]).count
280. mov ebx,sum
281. add ebx,eax
282. mov sum,ebx                             ;sum=sum+pmap->fre\*pmap->count;
284. mov edi,pmap
285. mov eax,(table ptr [edi]).link
286. mov pmap,eax                            ;pmap=pmap->link;
288. .endw
290. mov edx,offset output2
291. call WriteString
292. mov eax,sum
293. call WriteDec
294. call Crlf
295. ;printf("Minimum weighted external path length: %d\n",sum);
296. mov edx,offset output3
297. call WriteString
298. mov eax,cfre
299. mov ebx,8
300. mul ebx
301. sub eax,sum
302. call WriteDec
303. mov al,'/'
304. call WriteChar
305. mov eax,cfre
306. mov ebx,8
307. mul ebx
308. call WriteDec
309. call Crlf
310. call Crlf
311. mov edx,offset output4
312. call WriteString
313. ;printf("Compression ratio: %d/%d\n\nPlease input the code: ",cfre\*8-sum,cfre\*8);
315. mov eax,0
316. mov sum,eax             ;sum=0;
317. mov cnum,eax            ;int cnum=0;
318. mov edx,offset inputc2
319. mov ecx,Max
320. call ReadString
321. ;輸入需decode的table like 010101010101110101
322. mov esi,offset inputc2
323. .while byte ptr [esi]!=0
324. movzx eax,byte ptr[esi]
325. sub eax,'0'
326. mov ebx,sum
327. add ebx,sum
328. add ebx,eax
329. mov sum,ebx
330. ;sum=sum\*2+c-'0';
331. inc cnum
332. ;cnum++;
333. mov eax,map
334. mov pmap,eax
335. ;pmap=map;
337. .while pmap!=0
338. mov edi,pmap
339. mov eax,(table ptr [edi]).key
340. mov ebx,(table ptr [edi]).count
341. .if (sum==eax && cnum==ebx)
342. mov edi,pmap
343. mov al,byte ptr (table ptr [edi]).lab
344. call WriteChar
345. ;printf("%c",pmap->label);
346. mov eax,0
347. mov sum,eax
348. ;sum=0;
349. mov cnum,eax
350. ;cnum=0;
352. .break
353. .endif
354. mov edi,pmap
355. mov eax,(table ptr [edi]).link
356. mov pmap,eax
357. ;pmap=pmap->link;
358. .endw
359. inc esi
360. .endw
361. call Crlf       ;printf("\n");
362. call WaitMsg
363. exit
364. main ENDP
365. ;顯示table
366. print proc,p:dword
367. local nn:dword,ii:sdword
368. .while p!=0
369. mov edi,p
370. mov eax,(table ptr [edi]).lab
371. .if eax==' '
372. mov edx,offset output5
373. call WriteString
374. ;printf("Space: ");
375. .else
376. mov edi,p
377. mov al,byte ptr (table ptr [edi]).lab
378. call WriteChar
379. mov al,':'
380. call WriteChar
381. mov al,' '
382. call WriteChar
383. ;printf("%c: ",p->label);
384. .endif
386. mov edi,p
387. mov eax,(table ptr [edi]).key
388. mov nn,eax
389. ;int n=p->key;
391. ;int i;
392. mov edi,p
393. mov eax,(table ptr [edi]).count
394. dec eax
395. mov ii,eax
396. ;i=p->count-1
397. .while ii>=0
399. mov eax,nn
400. mov cl,byte ptr ii
401. shr eax,cl
402. call WriteDec
403. ;printf("%d",n/(int)pow(2,i));
404. mov ebx,1
405. mov cl,byte ptr ii
406. shl ebx,cl
407. mov edx,0
408. mov eax,nn
409. div ebx
410. mov nn,edx
411. ;n=n%(int)pow(2,i);
412. dec ii
413. .endw
414. call Crlf
415. ;printf("\n");
416. mov edi,p
417. mov eax,(table ptr [edi]).link
418. mov p,eax
419. ;p=p->link;
420. .endw
421. ret
422. print endp
423. ;遞迴指派0 1
424. searchleaf proc,rr:dword,tt:dword,vv:dword,cc:dword
425. local localtemp:dword,localtemp2:dword
426. mov edi,rr
427. mov eax,(htree ptr [edi]).left
428. mov ebx,(htree ptr [edi]).right
429. .if(eax==0 && ebx==0)
430. ;ptable temp;
431. invoke GetProcessHeap
432. mov hHeap,eax
433. invoke HeapAlloc,hHeap,HEAP\_ZERO\_MEMORY,type htree
434. mov localtemp,eax                       ;temp=(ptable)malloc(sizeof(struct table));
436. mov eax,vv
437. mov edi,localtemp
438. mov (table ptr [edi]).key,eax           ;temp->key=v;
440. mov eax,cc
441. mov edi,localtemp
442. mov (table ptr [edi]).count,eax         ;temp->count=c;
444. mov edi,rr
445. mov eax,(htree ptr [edi]).fre
446. mov edi,localtemp
447. mov (table ptr [edi]).fre,eax       ;temp->fre=r->fre;
449. mov edi,rr
450. mov eax,(htree ptr [edi]).lab
451. mov edi,localtemp
452. mov (table ptr [edi]).lab,eax       ;temp->label=r->lab;
454. mov edi,tt
455. mov esi,dword ptr [edi]
456. .if esi==0
457. mov eax,localtemp
458. mov edi,tt
459. mov dword ptr [edi],eax         ;\*t=temp;
460. .else
461. mov edi,tt
462. mov eax,dword ptr [edi]
463. mov localtemp2,eax              ;ptable temp2=\*t;
465. mov edi,localtemp2
466. mov eax,(table ptr [edi]).link
467. .while eax!=0
468. mov edi,localtemp2
469. mov eax,(table ptr [edi]).link
470. mov localtemp2,eax          ;temp2=temp2->link;
472. mov edi,localtemp2
473. mov eax,(table ptr [edi]).link
474. .endw
475. mov eax,localtemp
476. mov edi,localtemp2
477. mov (table ptr [edi]).link,eax  ;temp2->link=temp;
479. .endif
480. ret             ;return;
481. .endif
483. mov edi,rr
484. mov eax,(htree ptr [edi]).left
485. mov ebx,vv
486. add ebx,vv
487. mov ecx,cc
488. add ecx,1
489. invoke searchleaf,eax,tt,ebx,ecx;searchleaf(r->left,t,v\*2+0,c+1);
490. mov edi,rr
491. mov eax,(htree ptr [edi]).right
492. mov ebx,vv
493. add ebx,vv
494. add ebx,1
495. mov ecx,cc
496. add ecx,1
497. invoke searchleaf,eax,tt,ebx,ecx;searchleaf(r->right,t,v\*2+1,c+1);
498. ret
499. searchleaf endp
500. ;中序走訪並輸出
501. tprint proc,r:dword
502. .if r!=0
503. mov edi,r
504. mov eax,(htree ptr [edi]).left
505. invoke tprint,eax
507. mov edi,r
508. mov al,byte ptr (htree ptr [edi]).lab
509. call WriteChar
510. mov al,' '
511. call WriteChar
512. mov eax,(htree ptr [edi]).fre
513. call WriteDec
514. call Crlf
516. mov edi,r
517. mov eax,(htree ptr [edi]).right
518. invoke tprint,eax
519. .endif
520. ret
521. tprint endp
522. ;bubble sort
523. sort proc,f:dword
524. local count:dword,p:dword,i:dword,j:dword,after:dword,temp1:dword,temp2:dword
525. mov eax,0
526. mov count,eax                   ;int count=0;
527. mov edi,f
528. mov eax,dword ptr [edi]
529. mov p,eax                       ;ptree p=\*f;
531. .while p!=0
532. inc count                   ;count++;
533. mov edi,p
534. mov eax,(htree ptr [edi]).slink
535. mov p,eax                   ;p=p->slink;
536. .endw
537. mov eax,1
538. mov i,eax       ;i=1
539. mov eax,count
540. .while i<=eax
542. mov edi,f
543. mov eax,dword ptr [edi]
544. mov p,eax                       ;p=\*f;
546. mov eax,1           ;j=1
547. mov j,eax
548. mov eax,count
549. sub eax,i
550. .while j<=eax
551. mov edi,p
552. mov eax,(htree ptr [edi]).fre
553. mov ecx,(htree ptr [edi]).lab
554. mov esi,(htree ptr [edi]).slink
555. mov ebx,(htree ptr [esi]).fre
556. mov edx,(htree ptr [esi]).lab
558. .if  (eax > ebx) || ((eax==ebx) && (ecx > edx));(p->fre > p->slink->fre) || ((p->fre==p->slink->fre) && (p->lab > p->slink->lab))
559. mov edi,p
560. mov esi,(htree ptr [edi]).slink
561. mov after,esi           ;ptree after=p->slink;
563. mov edi,p
564. mov eax,(htree ptr [edi]).fre
565. mov temp1,eax           ;temp1=p->fre;
567. mov edi,after
568. mov eax,(htree ptr [edi]).fre
569. mov edi,p
570. mov (htree ptr [edi]).fre,eax;p->fre=after->fre;
572. mov eax,temp1
573. mov edi,after
574. mov (htree ptr [edi]).fre,eax;after->fre=temp1;
576. mov edi,p
577. mov eax,(htree ptr [edi]).lab
578. mov temp1,eax               ;temp1=p->lab;
580. mov edi,after
581. mov eax,(htree ptr [edi]).lab
582. mov edi,p
583. mov (htree ptr [edi]).lab,eax;p->lab=after->lab;
585. mov eax,temp1
586. mov edi,after
587. mov (htree ptr [edi]).lab,eax;after->lab=temp1;
589. mov edi,p
590. mov eax,(htree ptr [edi]).left
591. mov temp2,eax               ;temp2=p->left;
593. mov edi,after
594. mov eax,(htree ptr [edi]).left
595. mov edi,p
596. mov (htree ptr [edi]).left,eax;p->left=after->left;
598. mov eax,temp2
599. mov edi,after
600. mov (htree ptr [edi]).left,eax;after->left=temp2;
602. mov edi,p
603. mov eax,(htree ptr [edi]).right
604. mov temp2,eax                   ;temp2=p->right;
606. mov edi,after
607. mov eax,(htree ptr [edi]).right
608. mov edi,p
609. mov (htree ptr [edi]).right,eax;p->right=after->right;
611. mov eax,temp2
612. mov edi,after
613. mov (htree ptr [edi]).right,eax;after->right=temp2;
614. .endif
615. mov edi,p
616. mov esi,(htree ptr [edi]).slink
617. mov p,esi                       ;p=p->slink;
619. inc j           ;++j
620. mov eax,count
621. sub eax,i
622. .endw
623. inc i               ;++i
624. mov eax,count
625. .endw
626. ret
627. sort endp
628. ;為了讓輸出table照字母順序
629. tablesort proc,ff:dword
630. local count:dword,p:dword,i:dword,j:dword,after:dword,temp1:dword,temp2:dword
631. mov eax,0
632. mov count,eax                   ;int count=0;
633. mov edi,ff
634. mov eax,dword ptr [edi]
635. mov p,eax                       ;ptree p=\*f;
637. .while p!=0
638. inc count                   ;count++;
639. mov edi,p
640. mov eax,(table ptr [edi]).link
641. mov p,eax                   ;p=p->slink;
642. .endw
643. mov eax,1
644. mov i,eax       ;i=1
645. mov eax,count
646. .while i<=eax
648. mov edi,ff
649. mov eax,dword ptr [edi]
650. mov p,eax                       ;p=\*f;
652. mov eax,1           ;j=1
653. mov j,eax
654. mov eax,count
655. sub eax,i
656. .while j<=eax
658. mov edi,p
659. mov eax,(table ptr [edi]).lab
660. mov esi,(table ptr [edi]).link
661. mov ebx,(table ptr [esi]).lab
662. .if  eax>ebx;(p->label > p->link->label)
663. mov edi,p
664. mov esi,(table ptr [edi]).link
665. mov after,esi           ;ptree after=p->link;
667. mov edi,p
668. mov eax,(table ptr [edi]).fre
669. mov temp1,eax           ;temp1=p->fre;
671. mov edi,after
672. mov eax,(table ptr [edi]).fre
673. mov edi,p
674. mov (table ptr [edi]).fre,eax;p->fre=after->fre;
676. mov eax,temp1
677. mov edi,after
678. mov (table ptr [edi]).fre,eax;after->fre=temp1;
680. mov edi,p
681. mov eax,(table ptr [edi]).lab
682. mov temp1,eax               ;temp1=p->lab;
684. mov edi,after
685. mov eax,(table ptr [edi]).lab
686. mov edi,p
687. mov (table ptr [edi]).lab,eax;p->lab=after->lab;
689. mov eax,temp1
690. mov edi,after
691. mov (table ptr [edi]).lab,eax;after->lab=temp1;
693. mov edi,p
694. mov eax,(table ptr [edi]).key
695. mov temp2,eax               ;temp2=p->key;
697. mov edi,after
698. mov eax,(table ptr [edi]).key
699. mov edi,p
700. mov (table ptr [edi]).key,eax;p->key=after->key;
702. mov eax,temp2
703. mov edi,after
704. mov (table ptr [edi]).key,eax;after->key=temp2;
706. mov edi,p
707. mov eax,(table ptr [edi]).count
708. mov temp2,eax                   ;temp2=p->count;
710. mov edi,after
711. mov eax,(table ptr [edi]).count
712. mov edi,p
713. mov (table ptr [edi]).count,eax;p->count=after->count;
715. mov eax,temp2
716. mov edi,after
717. mov (table ptr [edi]).count,eax;after->count=temp2;
718. .endif
719. mov edi,p
720. mov esi,(table ptr [edi]).link
721. mov p,esi                       ;p=p->slink;
723. inc j           ;++j
724. mov eax,count
725. sub eax,i
726. .endw
727. inc i               ;++i
728. mov eax,count
729. .endw
730. ret
731. tablesort endp
732. END main

Readstring:讀取輸入在Cmd的字串

OpenInputFile ReadFromFile:用來讀txt檔案 在cmd中打入檔名以讀取

一開始將排除除了英文跟空格的字元 再來先創造一個新的連結藉此每個英文字將擁有頻率跟ascii字母序的結構

再來將這些節點連結成樹 在連結過程中會逕行排序 使得連結方法能像下圖

Sort:就是其中使用的排序 其為bubble sort

過後會使用tprint: 其為中序走訪 來檢查樹是否建對

Searchleaf:之後再將樹葉們賦予0 1這過程是用遞迴去做的

Tablesort:就把列表照字母順序排好

Prinit:輸出列表

再來顯示壓縮率跟全長

然後輸入需要decode的字串 若該code不足上面編碼的時候會省略

最後輸出對應編碼的字串