```c
#include <stdio.h>
#include "i2p.h"
// WordsHandling V5 - Applying modularity (the case of wordStats)
// Author: Kleanthis Thramboulidis
#define TERMINATE           0
#define ADD_WORD            1
#define DISPLAY_WORDS       2
#define INCR_SORT           3
#define DECR_SORT           4
#define WORD_STATS          5
#define CHAR_STATS          6
#define SEARCH_WORD         7

#define MAX_WORDS       6
#define MAX_WORD_LEN    10

int menu(void);
void addWord(void);
void displayWords(void);
void incSort(void);
void decSort(void);
void wordStats(void);
void charStats(char ch);
void searchWord(char str[]);
int charAp(char word[],char ch);
char getChar(char message[]);
void initializeWordStats(char str[]);
void updateWordStats(char str[]);
void displayWordStats(void);

//void initializeCharStats(char str[],char ch);
//void updateCharStats(char str[],char ch){
//void displayCharStats(void);

char words[MAX_WORDS][MAX_WORD_LEN]={"kleanth","alexand","zois","nikou"};
int freeEntry=4;

void main(){
    int selectedOp;
    char ch;
```

```
41:     char word[MAX_WORD_LEN];
42:
43:     printf("WordsHandling V5 - Applying modularity (the case
    of wordStats)\n\n");
44:     selectedOp=menu();
45:     while(selectedOp!= TERMINATE){
46:         switch(selectedOp){
47:             case ADD_WORD :
48:                 addWord();
49:                 break;
50:             case DISPLAY_WORDS :
51:                 displayWords();
52:                 break;
53:             case INCR_SORT :
54:                 incSort();
55:                 break;
56:             case DECR_SORT  :
57:                 decSort();
58:                 break;
59:             case WORD_STATS :
60:                 wordStats();
61:                 break;
62:             case CHAR_STATS :
63:                 ch = getChar("Dose xaraktira gia
    statistika:");
64:                 charStats(ch);
65:                 break;
66:             case SEARCH_WORD :
67:                 getStringV2("Dose lexi gia anazitisi:",word,
    MAX_WORD_LEN);
68:                 searchWord(word);
69:                 break;
70:             default :
71:                 printf("Operation is not supported\n");
72:                 break;
73:             }
74:         selectedOp=menu();
75:         }
76:     printf("WordsHandling V1 terminated\n");
77: }
78:
79: int menu(void){
```

```
 80:     int choice;
 81:
 82:     printf("\n\n-----------MENU--------------\n");
 83:     printf("0 - TERMINATE\n");
 84:     printf("1 - ADD WORD\n");
 85:     printf("2 - DISPLAY WORDS\n");
 86:     printf("3 - INCREMENTAL SORT\n");
 87:     printf("4 - DECREMENTAL SORT\n");
 88:     printf("5 - WORD STATISTICS\n");
 89:     printf("6 - CHARACTER STATISTICS\n");
 90:     printf("7 - SEARCH WORD \n-----------------------------
    \n");
 91:     printf("Select operation:");
 92:     scanf("%d",&choice);
 93:     fflush(stdin);
 94:     return (choice);
 95: }
 96:
 97: void addWord(void){
 98:     printf("add word\n");
 99:     getStringV2("Dose lexi:",words[freeEntry],9);
100:     freeEntry++;
101: }
102:
103: void displayWords(void){
104:     printf("---Words list------\n");
105:     for(int i=0;i<freeEntry;i++)
106:         printf("%s\n",words[i]);
107:     printf("-------------------\n");
108: }
109:
110: void incSort(void){
111:     printf("incremental sort\n");
112:     sortStringArrayInc(words,freeEntry, MAX_WORD_LEN);
113: }
114:
115: void decSort(void){
116:     printf("decremental sort\n");
117:     sortStringArrayDec(words,freeEntry, MAX_WORD_LEN);
118: }
119:
120: void searchWord(char str[]){
```

```
121:     printf("search word\n");
122:     for(int i=0;i<freeEntry;i++)
123:         if(strcmp(str,words[i])==0){
124:             printf("Word %s found at pos %d\n",str,i);
125:             return;
126:         }
127:     printf("Word %s not found\n",str);
128: }
129:
130: int curWord_len;
131: int max_len, min_len;
132: int total_len=0;
133:
134: void wordStats(void){
135: //  int curWord_len;
136: //  int max_len, min_len,i;
137: //  int total_len=0;
138:     int i;
139:
140: //  printf("word statistics\n");
141:     initializeWordStats(words[0]);
142: //  curWord_len = strlen(words[0]);
143: //  max_len = min_len = curWord_len;
144: //  total_len += curWord_len;
145:     for(i=1;i<freeEntry;i++){
146:         updateWordStats(words[i]);
147: //      curWord_len=strlen(words[i]);
148: //      total_len += curWord_len;
149: //      if(max_len<curWord_len)
150: //          max_len = curWord_len;
151: //      if(min_len>curWord_len)
152: //          min_len = curWord_len;
153:         }
154:     displayWordStats();
155: //  printf("max len = %d\n",max_len);
156: //  printf("min len = %d\n",min_len);
157: //  printf("average len = %f\n",
    freeEntry==0?0:(float)total_len/freeEntry);
158: }
159:
160: void initializeWordStats(char str[]){
161:     curWord_len = strlen(str);
```

```
162:     max_len = min_len = curWord_len;
163:     total_len += curWord_len;
164: }
165:
166: void updateWordStats(char str[]){
167:     curWord_len=strlen(str);
168:     total_len += curWord_len;
169:     if(max_len<curWord_len)
170:         max_len = curWord_len;
171:     if(min_len>curWord_len)
172:         min_len = curWord_len;
173: }
174:
175: void displayWordStats(void){
176:     printf("max len = %d\n",max_len);
177:     printf("min len = %d\n",min_len);
178:     printf("average len = %f\n",
     freeEntry==0?0:(float)total_len/freeEntry);
179: }
180:
181: char getChar(char message[]){
182:     char ch;
183:     fflush(stdin);
184:     printf(message);
185:     ch = getchar();
186:     fflush(stdin);
187:     return ch;
188: }
189:
190: void charStats(char ch){
191:     printf("character statistics\n");
192: }
193:
194: int charAp(char word[],char ch){
195:     int i=0;
196:     int count = 0;
197:
198:     while(word[i]!= '\0')
199:         word[i++]== ch? count++ : count;
200:     return count;
201: }
202:
```

203:
204: