```
1
   2
   /*HW04 Furkan Erdol 131044065 part3.c
3
4
                                                        */
   /*Written by Furkan Erdol on March 12, 2015
5
   /*Description
6
7
8
                                                        */
   /*>>>Learns XUniversity's encoding system from given files, decodes their
9
   /*encoded messages and writes as plain text to a file<---
                                                        */
10
                                                        */
11
   /*Inputs:
                                                        */
12
   /* -Character list
   /* -Sample
                                                        */
13
14
   /* -X University encoded message
15
   /*Outputs:
16
   /* -X University message
17
18
   /*Points: 49
19
20
                        Includes
21
22
   #include <stdio.h>
23
24
   #define TRUE 1
25
   #define FALSE 0
   #define CHARACTERFILE "Files/Q3/CharacterList.txt"
26
   #define SAMPLEFILE "Files/Q3/Sample.txt"
27
   #define ENCODEDFILE "Files/Q3/XUniversityEncoded.txt"
28
29
   #define PLAINTEXTFILE "Files/Q3/XUniversityMessage.txt"
30
   31
32
    * Swaps values of two integers
33
   34
   void swap_int(int *a, int *b);
35
   36
37
    * Swaps values of two characters
   38
39
   void swap_char(char *a, char *b);
40
   41
42
    * Sorts characters according to counts.
   43
44
   void sort(char *a, int a_num, char *b, int b_num, char *c, int c_num);
45
46
   47
    * Check whether character is big ASCII letter or not return TRUE or FALSE
   48
49
   int is letter(char c);
50
51
   52
    * Read characters from character list file and if character is letter assign *
53
    st characters to c1, c2 and c3. If file has not three letters assign NULL to \, ^*
54
    * input char by order. For example file has two letters assign proper letters*
55
    st to c1 and c2 and assign NULL to c3. If file has four letters assign c1, c2 st
56
    * and c3 first three letters. Return number of letters in character list file*
57
    58
   int read_character_list(FILE* f_in_ptr, char *c1, char *c2, char *c3);
59
   60
61
    st Read letters from Sample file and compute frequency of letters. Then sort \, ^*
62
    st it inside this function. Call sort function. At the end make sure that stc1 st
63
    * keeps most frequent used letter, *c3 keeps least frequent used letter
64
    * and *c2 keeps remained letter
   65
66
   void count_letters(FILE *f_in_ptr, char *c1, char *c2, char *c3);
67
   68
    * Read from XUniversityEncoded file to decode message and write decoded
69
70
    * (plain text) message to XUniversityMessage file. Make sure c1 keeps most
71
    * frequent used letter, c3 keeps least frequent used letter and c2 keeps
72
    * remained letter while calling function. According to frequency you know
```

```
73
       * their codes. c1: 0, c2: 10, c3: 110.
                                             **************
 74
 75
      void decode(FILE *f in ptr, FILE *f out ptr, char c1, char c2, char c3);
 76
 77
 78
 79
      int
 80
      main(void)
 81
      {
 82
 83
          int character_number; /* Number of characters read */
 84
                               /* Letters of messaging system's alphabet */
          char c1, c2, c3;
 85
          FILE *f_character_list_ptr, /* CharacterList.txt */
 86
 87
                                    /* Sample.txt */
               *f_sample_file_ptr,
 88
                                      /* XUniversityEncoded.txt */
               *f_encoded_ptr,
 89
               *f_plain_text_ptr;
                                      /* XUniversityMessage.txt */
 90
 91
 92
          /* Open character list file */
 93
          f_character_list_ptr=fopen(CHARACTERFILE, "r");
 94
 95
          /*Exit program and print error if character list file could not be opened to read */
 96
          if(f_character_list_ptr==NULL)
 97
 98
              printf("CharacterList.txt couldn't open...\n");
 99
              return 0:
100
          }
101
102
          /* Read letters of messaging system's alphabet and get return value of number of letter */
103
          character_number=read_character_list(f_character_list_ptr, &c1, &c2, &c3);
104
105
          /*if number of letter read is not equal to three exit program */
106
          if(character_number!=3)
107
          {
108
              printf("Warning: Number of letter not equal to three...\n");
109
110
          }
111
112
          /* Close character list */
113
          fclose(f_character_list_ptr);
114
115
          /* Open sample file */
116
          f_sample_file_ptr=fopen(SAMPLEFILE, "r");
117
118
          /* Exit program and print error if sample file could not be opened to read */
119
          if(f_sample_file_ptr==NULL)
120
          {
121
              printf("Sample.txt couldn't open...\n");
122
              return 0;
123
          }
124
125
          /* Read letters from Sample file and compute frequency of letters. Then
126
           * sort it inside this function
127
          count_letters(f_sample_file_ptr, &c1, &c2, &c3);
128
129
          /* Close sample file */
130
          fclose(f_sample_file_ptr);
131
132
          /* Open encoded file and plain text file */
133
          f_encoded_ptr=fopen(ENCODEDFILE, "r");
134
          f_plain_text_ptr=fopen(PLAINTEXTFILE, "w");
135
136
          /*Exit program and print error if encoded file could not be opened to read */
137
          if(f_encoded_ptr==NULL)
138
          {
139
              printf("XUniversityEncoded.txt couldn't open...\n");
140
              return 0:
141
          }
142
143
          /*Exit program and print error if plain text file could not be opened to write */
144
          if(f_plain_text_ptr==NULL)
```

```
145
        {
146
           printf("XUniversityMessage.txt couldn't open...\n");
147
           return 0;
148
        }
149
150
        /* Read from XUniversityEncoded file to decode message and write decoded
         * (plain text) message to XUniversityMessage file
151
152
        decode(f_encoded_ptr, f_plain_text_ptr, c1, c2, c3);
153
154
        /* Close encoded file and plain text file */
155
        fclose(f_encoded_ptr);
156
        fclose(f_plain_text_ptr);
157
158
        return 0;
159
160
161
    }
162
163
     164
165
     * Swaps values of two integers
     166
167
    void swap_int(int *a, int *b)
168
169
170
        int temp;
171
        temp=*a;
172
173
174
        *a=*b;
175
        *b=temp;
176
177
     }
178
179
180
     * Swaps values of two characters
                              181
182
     void swap_char(char *a, char *b)
183
184
185
        char temp; /* Temporary variable */
186
187
        temp=*a;
188
189
        *a=*b;
190
        *b=temp;
191
     }
192
     193
194
     * Sorts characters according to counts.
                                    195
196
     void sort(char *a, int a num, char *b, int b num, char *c, int c num)
197
198
199
        if(a_num<b_num)</pre>
200
201
           swap_char(a, b);
202
           swap_int(&a_num, &b_num);
203
204
        if(a_num<c_num)</pre>
205
206
           swap_char(a, c);
207
           swap_int(&a_num, &c_num);
208
        if(b_num<c_num)</pre>
209
210
        {
211
           swap_char(b, c);
212
           swap_int(&b_num, &c_num);
213
        }
214
    }
215
216
```

```
217
      * Check whether character is big ASCII letter or not return TRUE or FALSE
218
219
     int is letter(char c)
220
221
222
         if(c>=(int)'A'&&c<=(int)'Z')
223
             return TRUE;
224
         el se
225
             return FALSE;
226
227
     }
228
229
     230
231
      * Read characters from character list file and if character is letter assign *
232
      ^st characters to c1, c2 and c3. If file has not three letters assign NULL to ^st
233
      * input char by order. For example file has two letters assign proper letters*
234
      st to c1 and c2 and assign NULL to c3. If file has four letters assign c1, c2 st
235
      * and c3 first three letters. Return number of letters in character list file*
236
237
     int read_character_list(FILE* f_in_ptr, char *c1, char *c2, char *c3)
238
239
         char character; /* Read character variable */
240
241
         int counter = 0; /* Number of letters in file */
242
243
244
         while(fscanf(f_in_ptr, "%c", &character)!=EOF) /* Reads the file until the end */
245
246
             if(is_letter(character)==TRUE) /* Call is_letter function */
247
248
                 counter++;
249
250
                 if(counter==1) /* The first letter is assigned to c1 */
251
                    *c1=character:
252
                 else if(counter==2) /* The second letter is assigned to c2 */
253
254
                 else if(counter==3) /* The third letter is assigned to c3 */
255
                    *c3=character;
256
             }
257
         }
258
259
260
         return counter;
261
262
      263
264
      * Read letters from Sample file and compute frequency of letters. Then sort
265
      * it inside this function. Call sort function. At the end make sure that *c1 *
266
      * keeps most frequent used letter, *c3 keeps least frequent used letter
267
      * and *c2 keeps remained letter
      268
269
     void count_letters(FILE *f_in_ptr, char *c1, char *c2, char *c3)
270
271
         int count_c1=0, /* Counts frequency of letter c1 */
272
             count_c2=0, /* Counts frequency of letter c2 */
273
             count_c3=0; /* Counts frequency of letter c3 */
274
275
         char character; /* Read character variable */
276
277
         \mbox{while}(\mbox{fscanf}(\mbox{f\_in\_ptr}, \mbox{"%c"}, \mbox{\&character})!=\mbox{EOF}) /* \mbox{Reads the file until the end */
278
279
             if(character==*c1)
280
                 count_c1++;
281
             else if(character==*c2)
282
                count c2++;
283
             else if(character==*c3)
284
                 count_c3++;
285
         }
286
287
         sort(c1, count_c1, c2, count_c2, c3, count_c3); /* Call short function */
288
```

```
289
     290
291
      * Read from XUniversityEncoded file to decode message and write decoded
292
      ^{st} (plain text) message to XUniversityMessage file. Make sure c1 keeps most
293
      * frequent used letter, c3 keeps least frequent used letter and c2 keeps
294
      * remained letter while calling function. According to frequency you know
295
      * their codes. c1: 0, c2: 10, c3: 110.
296
297
     void decode(FILE *f_in_ptr, FILE *f_out_ptr, char c1, char c2, char c3)
298
299
        char character; /* Read character variable */
300
301
        int count_number_of_1=0; /* Counts number of 1 */
302
303
304
        while(fscanf(f_in_ptr, "%c", &character)!=EOF) /* Reads the file until the end */
305
306
307
           if(character=='1')
308
           count_number_of_1++;
309
           else
310
           {
311
312
               switch(count_number_of_1){
313
314
               case 0 : fprintf(f_out_ptr, "%c", c1);
315
                      break;
316
               case 1 : fprintf(f_out_ptr, "%c", c2);
317
                      break;
318
               case 2 : fprintf(f_out_ptr, "%c", c3);
319
                      break;
320
               }
321
322
           count_number_of_1=0;
323
324
           }
325
326
        }
327
328
     }
329
330
     331
                    End of HW04 Furkan Erdol 131044065 part3.c
332
```