```
/*****************************************************************************************************************
*
 * Program Name:        Caesar Encoder/Decoder v007 (ccipher.exe)
 * Course:              CIS-61, C++ Language Programming
 * Instructor:          C. Polen
 * Project:             Assignment 6
 * Created Date:        November 01st, 2010
 * Due Date:            December 03rd, 2010
 * Created By:          Chad Philip Johnson
 * Purpose:             Encode and decode messages consisting of uppercase and lowercase letters
 * Editor/IDE:          Notepad++
 * Resoluton:           1024x768
 * Compiler:            MinGW C++
 * Acknowledgements:    None
*****************************************************************************************************************
*/

#include <iostream>
using namespace std;

class SecretString
    {
    private:

        // change value for maximum user input length
        static const int    MAXSTRINGLENGTH = 100;
        // change value to discard extra input (double maximum input length is safe)
        static const int    MAXIGNORELENGTH = 200;

        char                chrInputPhrase[MAXSTRINGLENGTH], chrModifiedPhrase[MAXSTRINGLENGTH];
        char                chrUserResponse;

    public:


        /*****************************************************************************************************************
        *
        * Function Name:        fncUserPrompt
        * Parameters:           None
        * Return Value:         char
        * Purpose:              Ask user which portion of the program he/she would like to use (or quit)

        *****************************************************************************************************************
        */
        char fncUserPrompt()
            {

            // program title
            cout << "Caesar Encoder/Decoder v007" << endl << endl;

```

```
45              do {
46                  cout << "Would you like to (E)ncode, (D)ecode or (Q)uit?  ";
47                  cin >> chrUserResponse;
48                  cin.ignore();
49
50                  cout << endl;
51              // loop until user enters one of the six acceptable input values
52              } while (chrUserResponse != 'E' && chrUserResponse != 'e' && chrUserResponse != 'D' && chrUserResponse != 'd' &&
53                      chrUserResponse != 'Q' && chrUserResponse != 'q');
54
55              cout << endl << endl;
56
57              // return user value to calling function
58              return chrUserResponse;
59
60              }
61
62
        /*****************************************************************************************************************
        *
63      * Function Name:        fncEncode
64      * Parameters:           None
65      * Return Value:         void
66      * Purpose:              Ask user for string to be ENCODED (textual differences only between fncEncode and fncDecode)
67
        *****************************************************************************************************************
        */
68      void fncEncode()
69          {
70
71          // program title and current mode
72          cout << "Caesar v007 Encoder Mode" << endl << endl;
73
74          // prompt user for message to be encoded
75          // instructions adhere to program boundaries by reporting the MAXSTRINGLENGTH constant
76          cout << "Please enter the string you would like to encode (up to " << MAXSTRINGLENGTH << " characters)." << endl;
77          cout << "NOTE:  End your string with a ~ symbol and then press the enter key." << endl << endl;
78          // maximum number of characters inputted is dependant upon the program's MAXSTRINGLENGTH constant
79          cin.get(chrInputPhrase, MAXSTRINGLENGTH, '~');
80          // ignore MAXIGNORELENGTH constant excess entries and until newline
81          cin.ignore(MAXIGNORELENGTH, '\n');
82
83          cout << endl << endl;
84
85          }
86
87
        /*****************************************************************************************************************
        *
88      * Function Name:        fncDecode
```

```
89          * Parameters:           None
90          * Return Value:        void
91          * Purpose:             Ask user for message to be DECODED (textual differences only between fncEncode and fncDecode)
92
            ********************************************************************************************************************
            */
93          void fncDecode()
94              {
95
96              // program title and current mode
97              cout << "Caesar v007 Decoder Mode" << endl << endl;
98
99              // prompt user for message to be encoded
100             // instructions adhere to program boundaries by reporting the MAXSTRINGLENGTH constant
101             cout << "Please enter the string you would like to decode (up to " << MAXSTRINGLENGTH << " characters)." << endl;
102             cout << "NOTE:  End your string with a ~ symbol and then press the enter key." << endl << endl;
103             // maximum number of characters inputted is dependant upon the program's MAXSTRINGLENGTH constant
104             cin.get(chrInputPhrase, MAXSTRINGLENGTH, '~');
105             // ignore MAXIGNORELENGTH constant excess entries and until newline
106             cin.ignore(MAXIGNORELENGTH, '\n');
107
108             cout << endl << endl;
109
110             }
111
112
            /********************************************************************************************************************
            *
113         * Function Name:        fncCipher
114         * Parameters:           None
115         * Return Value:         void
116         * Purpose:              Prompt user for alphabet cipher shift value and perform safe calculations
117
            ********************************************************************************************************************
            */
118         void fncCipher()    {
119
120             int intCount = 0;
121             int intShift = 0;
122             // temporary location for chrInputPhrase while performing calculations
123             // some calculations produce values higher than 127 which produces bad values
124             // unsigned char arrays produce errors when used with cin.get()
125             int intTemp[MAXSTRINGLENGTH];
126
127             do {
128                 // prompt user for cipher shift value
129                 cout << "What integer cipher shift key will you use? (0-20)  " << endl;
130                 cin >> intShift;
131                 cin.ignore();
132
```

```
133                 cout << endl << endl;
134
135             // loop until entered value is between 0 and 20
136             } while (intShift < 0 || intShift > 20);
137
138             // user has selected ENCODER mode
139             if (chrUserResponse == 'E' || chrUserResponse == 'e')   {
140
141                 // loop until end of string is encountered
142                 while (chrInputPhrase[intCount] != '\0')    {
143
144                     // copy inputted string into integer array (to increase max value beyond 127)
145                     intTemp[intCount] = chrInputPhrase[intCount];
146
147                     // for all characters with integer values between 65 (A) - 90 (Z) and between 97 (a) - 122 (z)
148                     if ( (intTemp[intCount] >= 65 && intTemp[intCount] <= 90) ||
149                             (intTemp[intCount] >= 97 && intTemp[intCount] <= 122) )
150                     {
151
152                         // apply and assign cipher shift (ADDITION)
153                         intTemp[intCount] = intTemp[intCount] + intShift;
154
155                         // wrapper for A - Z characters, to prevent values outside of 65 - 90 scope
156                         // verify current intTemp value against chrInputPhrase to ensure it is within the 65 - 90 scope
157                         if (intTemp[intCount] > 90 && chrInputPhrase[intCount] <= 90)   {
158                             // wrap back to 'A' when values are greater than 90
159                             intTemp[intCount] = (intTemp[intCount] - 90) + 64;
160                         }
161
162                         // wrapper for a - z characters, to prevent values outside of 97 - 122 scope
163                         // NOTE: all remaining values at this point in the program are not a part of the 65 - 90 scope
164                         // NOTE: uncomment code below to verify current intTemp value against chrInputPhrase (ensures it is within
165                         //          the 97 - 122 scope)
166                         else if (intTemp[intCount] > 122 /* && chrInputPhrase[intCount] >= 97 */)   {
167                             // wrap back to 'a' when values are greater than 122
168                             intTemp[intCount] = (intTemp[intCount] - 122) + 96;
169                         }
170                     }
171
172                     // assign current value from intTemp array to chrModifiedPhrase array
173                     chrModifiedPhrase[intCount] = intTemp[intCount];
174
175                     intCount++;
176
177                 }
178             }
179
180             // user has selected DECODER mode
181             if (chrUserResponse == 'D' || chrUserResponse == 'd')   {
182
```

```
183                 // loop until end of string is encountered
184                 while (chrInputPhrase[intCount] != '\0')    {
185 
186                     // copy inputted string into integer array (to increase max value beyond 127)
187                     intTemp[intCount] = chrInputPhrase[intCount];
188 
189                     // for all characters with integer values between 65 (A) - 90 (Z) and between 97 (a) - 122 (z)
190                     if ( (intTemp[intCount] >= 65 && intTemp[intCount] <= 90) ||
191                             (intTemp[intCount] >= 97 && intTemp[intCount] <= 122) )
192                     {
193 
194                         // apply and assign cipher shift (SUBTRACTION)
195                         intTemp[intCount] = intTemp[intCount] - intShift;
196 
197                         // wrapper for a - z characters, to prevent values outside of 97 - 122 scope
198                         // verify current intTemp value against chrInputPhrase to ensure it is within the 97 - 122 scope
199                         if (intTemp[intCount] < 97 && chrInputPhrase[intCount] >= 97)   {
200                             // wrap back to 'a' when values are greater than 122
201                             intTemp[intCount] = 122 - (96 - intTemp[intCount]);
202                         }
203 
204                         // wrapper for A - Z characters, to prevent values outside of 65 - 90 scope
205                         // NOTE: all remaining values at this point in the program are not a part of the 97 - 122 scope
206                         // NOTE: uncomment code below to verify current intTemp value against chrInputPhrase (ensures it is within
207                         //          the 65 - 90 scope)
208                         else if (intTemp[intCount] < 65 /* && chrInputPhrase[intCount] <= 90 */)    {
209                             intTemp[intCount] = 90 - (64 - intTemp[intCount]);
210                         }
211 
212                     }
213 
214                     // assign current value from intTemp array to chrModifiedPhrase array
215                     chrModifiedPhrase[intCount] = intTemp[intCount];
216 
217                     intCount++;
218 
219                 }
220             }
221 
222             // apply null string value to last position of chrModifiedPhrase array
223             chrModifiedPhrase[intCount] = '\0';
224         }
225 
226 
        /*****************************************************************************************************
        *
227     * Function Name:        fncPrintMessage
228     * Parameters:           None
229     * Return Value:         void
230     * Purpose:              Print both the inputted and ciphered messages to the user; display firm warning
```

```
231
        ********************************************************************************************************************
        */
232         void fncPrintMessage()
233             {
234
235             // output original message
236             cout << "The original message...  " << endl;
237             cout << chrInputPhrase << endl << endl;
238
239             // output ciphered message
240             cout << "Has been ciphered into the following message...  " << endl;
241             cout << chrModifiedPhrase << endl << endl << endl << endl << endl << endl;
242
243             // firm warning
244             cout << "Please write it down and store it in a very safe place!" << endl;
245             cout << "Make sure you don't tell ANYBODY!  This is top secret stuff!" << endl << endl;
246
247             }
248
249     };
250
251 int main()
252     {
253
254     // create object and declare functions and variables
255     SecretString Caesar01;
256     char fncProgramOperation(SecretString&);
257
258     system("cls");
259
260     // loop through program until user quits
261     // value is returned as boolean TRUE/FALSE by called function fncProgramOperation()
262     // send object Caesar01
263     while ( fncProgramOperation(Caesar01) ) {
264
265         // cipher calculations
266         Caesar01.fncCipher();
267         system("cls");
268         // print results
269         Caesar01.fncPrintMessage();
270         system("pause");
271
272     }
273
274     // clear screen and end program
275     system("cls");
276     return 0;
277
278     }
```

```
279
280 /*****************************************************************************************************************
    *
281 * Function Name:        fncProgramOperation
282 * Parameters:           None
283 * Return Value:         boolean
284 * Purpose:              Allow user to choose which mode of the program to use (encoder or decoder) or to quit
285 *****************************************************************************************************************
    */
286 bool fncProgramOperation(SecretString& clsSecretStringTemp)
287     {
288
289     char chrCommand;
290
291     system("cls");
292
293     // assign returned value from class member function (also assigned to member variable chrUserResponse by member function)
294     chrCommand = clsSecretStringTemp.fncUserPrompt();
295
296     system("cls");
297
298     switch(chrCommand)
299         {
300         case 'E':
301         case 'e':
302             // when user selects 'e' or 'E' for ENCODE, call member function fncEncode
303             clsSecretStringTemp.fncEncode();
304             // continue program, TRUE
305             return true;
306
307         case 'D':
308         case 'd':
309             // when user selects 'd' or 'D' for DECODE, call member function fncDecode
310             clsSecretStringTemp.fncDecode();
311             // continue program, TRUE
312             return true;
313
314         case 'Q':
315         case 'q':
316             // when user selects 'q' or 'Q' for QUIT
317             // continue program, FALSE
318             return false;
319
320         }
321
322     }
323
```