```cpp

// COS30008, Midterm, Problem 2, 2024

#include "Vigenere.h"

#include <cctype>

using namespace std;

Vigenere::Vigenere(const string& aKeyword) :
    // Member initializers
    fKeyword(aKeyword),
    fKeywordProvider(aKeyword)
{
    initializeTable();
}

string Vigenere::getCurrentKeyword()
{
    string lResult;

    // Go through the keyword provider
    // and copy keyword characters into result string
    for (size_t i = 0; i < fKeyword.length(); i++)
    {
        char lChar = *fKeywordProvider;
        lResult += lChar;
        // Advance to next keyword character
        // while keeping the keyword provider unchanged
        fKeywordProvider << lChar;
    }

    return lResult;
}

void Vigenere::reset()
{
    fKeywordProvider.initialize(fKeyword);
}

char Vigenere::encode(char aCharacter)
{
    char lResult = aCharacter;

    // Only encode letters
    if (isalpha(aCharacter))
    {
        // Get encoded character from mapping table
        // Please ignore the warning about reading invalid data
```

```
50          // because we know that fKeywordProvider only contains letters
51          lResult = fMappingTable[*fKeywordProvider - 'A'][toupper
              (aCharacter) - 'A'];
52          // Advance to next keyword character
53          // and update keyword provider
54          fKeywordProvider << aCharacter;
55
56          // Keep the case of the original character
57          if (islower(aCharacter))
58          {
59              lResult = tolower(lResult);
60          }
61      }
62
63      return lResult;
64  }
65
66  char Vigenere::decode(char aCharacter)
67  {
68      char lResult = aCharacter;
69
70      if (isalpha(aCharacter))
71      {
72          char lRow = *fKeywordProvider - 'A';
73
74          // Find the column in the mapping table
75          for (size_t i = 0; i < CHARACTERS; i++)
76          {
77              if (fMappingTable[lRow][i] == toupper(aCharacter))
78              {
79                  lResult = 'A' + i;
80                  break;
81              }
82          }
83
84          // Advance to next keyword character
85          fKeywordProvider << lResult;
86
87          // Keep the case of the original character
88          if (islower(aCharacter))
89          {
90              lResult = tolower(lResult);
91          }
92      }
93
94      return lResult;
95  }
```