【C++】Day48

Class	C++
Date	@January 31, 2022
Material	
Series Number	
Summary	

【Ch11】Associative Container

11.3.5 A Word Transformation Map

We'll close this section with a program to illustrate creating, searching, and iterating across a map.

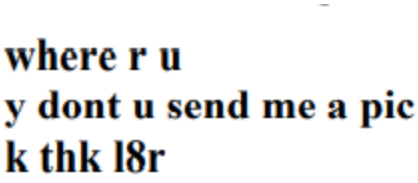
We'll write a program that, given one string, transforms it into another. The input to our program is two files. The first file contains rules that we will use to transform the text in the second file. Each rule consists of a words that might be in the input file and a phrase to use in its place.

The idea is that whenever the first words appears in the input, we will replace it with the corresponding phrase. The second file contains the text to transform.

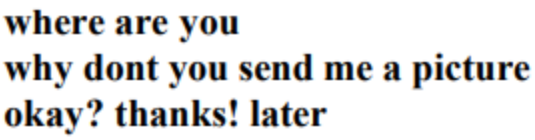
If the contents of the word-transformation file are:

```
brb be right back
k okay?
y why
r are
u you
pic picture
thk thanks!
l8r later
```

The given key file is:



then the program should generate the following output:



The Word Transformation Program

Our solution will use three functions:

- The `word_transform` function will manage the overall processing. It will take two `ifstream` arguments: The first will be bound to the word-transformation file and the second to the file of text we're to transform.
- The `buildMap` function will read the file of transformation rules and create a map from each word to its transformation.
- The `transform` function will take a string and return the transformation if there is one.

We'll start with the `word_transform` function:

```
void word_transform(ifstream &map_file, ifstream &input_file) {
    //build the map
    map<string, string> trans_map = buildMap(map_file);
    //Get each line by using the getline function
    string oneLine;
    while(std::getline(input_file, oneLine)) {
        //Create an istringstream object, linked to input_file
        std::istringstream stream(oneLine);
        string word;
        bool firstWord = true;
        //process each word
```

```
        while(stream >> word) {
            if(firstWord)
                firstWord = false;
            else
                std::cout << " ";
            //Call the transform function
            std::cout << transform(trans_map, word);
        }
        //Start a new line
        std::cout << std::endl;
    }
}
```

Then the `build_Map` function:

```
//Build the map from the given file
map<string, string> buildMap(ifstream &map_file) {
    map<string, string> trans_map;
    string key, oneLine;
    while(map_file >> key && getline(map_file, oneLine)) {
        //If oneLine contains elements more than just a space, skip the first space
        if(oneLine.size() > 1)
            trans_map[key] = oneLine.substr(1);
        else
            throw std::runtime_error("Cannot find value for " + key);
    }
    return trans_map;
}
```

Finally the transform function:

```
const string &transform(map<string, string> &trans_map, const string &word) {
    auto map_pos = trans_map.find(word);
    if(map_pos != trans_map.cend())
        return map_pos->second;
    return word;
}
```

Exercise

Exercise 11.33: Implement your own version of the word-transformation program.

See 11_33.cpp for code