

## - Exercise

In this lesson, we'll execute and analyze the given code.

### WE'LL COVER THE FOLLOWING ^

- Try it out

## Try it out #

The program how frequently a word is used in a text. This is a typical use-case for an ordered associative container.

- Execute the program
- Analyze the source code

 Test.rtf  

main.cpp



Test.rtf

```
#include <regex>
#include <algorithm>
#include <cstdlib>
#include <fstream>
#include <iostream>
#include <sstream>
#include <string>
#include <map>
#include <unordered_map>
#include <utility>

using str2Int = std::unordered_map<std::string, size_t>;
using intAndWords = std::pair<std::size_t, std::vector<std::string>>;
using int2Words = std::map<std::size_t, std::vector<std::string>>;

// count the frequency of each word
str2Int wordCount(const std::string& text){
    std::regex wordReg(R"(\w+)");
    std::sregex_iterator wordItBegin(text.begin(), text.end(), wordReg);
```

```

std::sregex_iterator wordItBegin(text.begin(), text.end(), wordReg);
const std::sregex_iterator wordItEnd;
str2Int allWords;
for (; wordItBegin != wordItEnd; ++wordItBegin){
    ++allWords[wordItBegin->str()];
}
return allWords;
}

// get to all frequencies the appropriate words
int2Words frequencyOfWords(str2Int& wordCount){
    int2Words freq2Words;
    for ( auto wordIt: wordCount ){
        auto freqWord= wordIt.second;
        if ( freq2Words.find(freqWord) == freq2Words.end() ){
            freq2Words.insert( intAndWords(freqWord, {wordIt.first} ));
        }
        else {
            freq2Words[freqWord].push_back(wordIt.first);
        }
    }
    return freq2Words;
}

int main(int argc, char* argv[]){

    std::cout << std::endl;

    // get the filename
    std::string myFile = "Test.rtf";

    // open the file
    std::ifstream file(myFile, std::ios::in);
    if (!file){
        std::cerr << "Can't open file "+ myFile + "!" << std::endl;
        exit(EXIT_FAILURE);
    }

    // read the file
    std::stringstream buffer;
    buffer << file.rdbuf();
    std::string text(buffer.str());

    // get the frequency of each word
    auto allWords= wordCount(text);

    std::cout << "The first 20 (key, value)-pairs: " << std::endl;
    auto end= allWords.begin();
    std::advance(end, 20);
    for (auto pair= allWords.begin(); pair != end; ++pair){
        std::cout << "(" << pair->first << ": " << pair->second << ")";
    }
    std::cout << "\n\n";

    std::cout << "allWords[Web]: " << allWords["Web"] << std::endl;
    std::cout << "allWords[The]: " << allWords["The"] << "\n\n";

    std::cout << "Number of unique words: ";
    std::cout << allWords.size() << "\n\n";

    size_t sumWords=0;
    for (auto wordIt: allWords) sumWords+= wordIt.second;
    std::cout << "Total number of words: " << sumWords << "\n\n";
}

```

```

auto allFreq= frequencyOfWords(allWords);

std::cout << "Number of different frequencies: " << allFreq.size() << "\n\n";

std::cout << "All frequencies: ";
for (auto freqIt: allFreq) std::cout << freqIt.first << " ";
std::cout << "\n\n";

std::cout << "The most frequently occurring word(s): " << std::endl;
auto atTheEnd= allFreq.rbegin();
std::cout << atTheEnd->first << " :";
for (auto word: atTheEnd->second) std::cout << word << " ";
std::cout << "\n\n";

std::cout << "All word which appears more then 1000 times:" << std::endl;
auto biggerIt= std::find_if(allFreq.begin(), allFreq.end(), [](intAndWords iAndW){return iAndW.second.size() > 1000;});
if (biggerIt == allFreq.end()){
    std::cerr << "No word appears more then 1000 times !" << std::endl;
    exit(EXIT_FAILURE);
}
else{
    for (auto allFreqIt= biggerIt; allFreqIt != allFreq.end(); ++allFreqIt){
        std::cout << allFreqIt->first << " :";
        for (auto word: allFreqIt->second) std::cout << word << " ";
        std::cout << std::endl;
    }
}
std::cout << std::endl;
}

```



In this chapter, we have studied ordered associative containers. In the next chapter, we'll study unordered associative containers in detail.