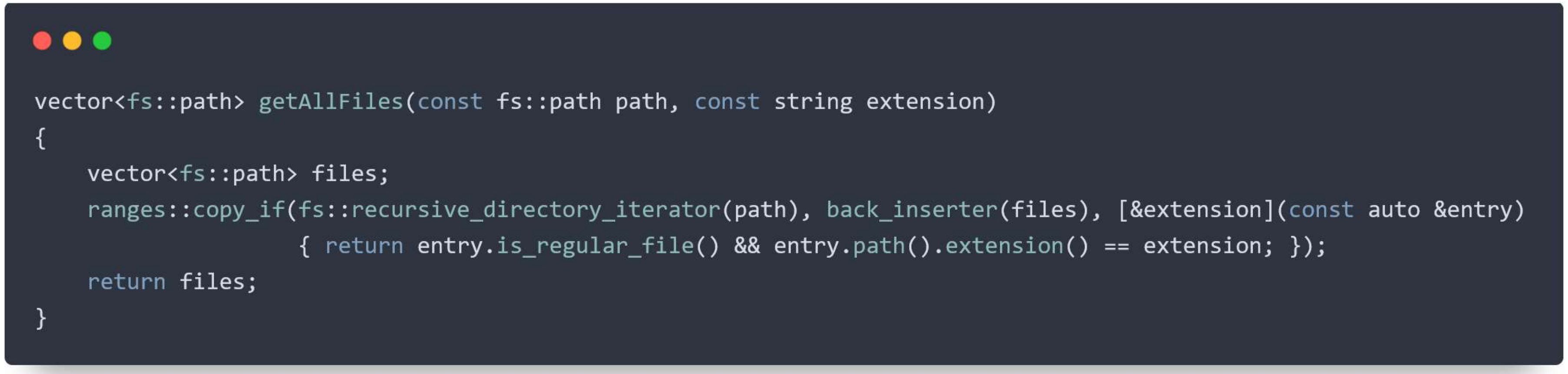
Word Count
Functional implementation in

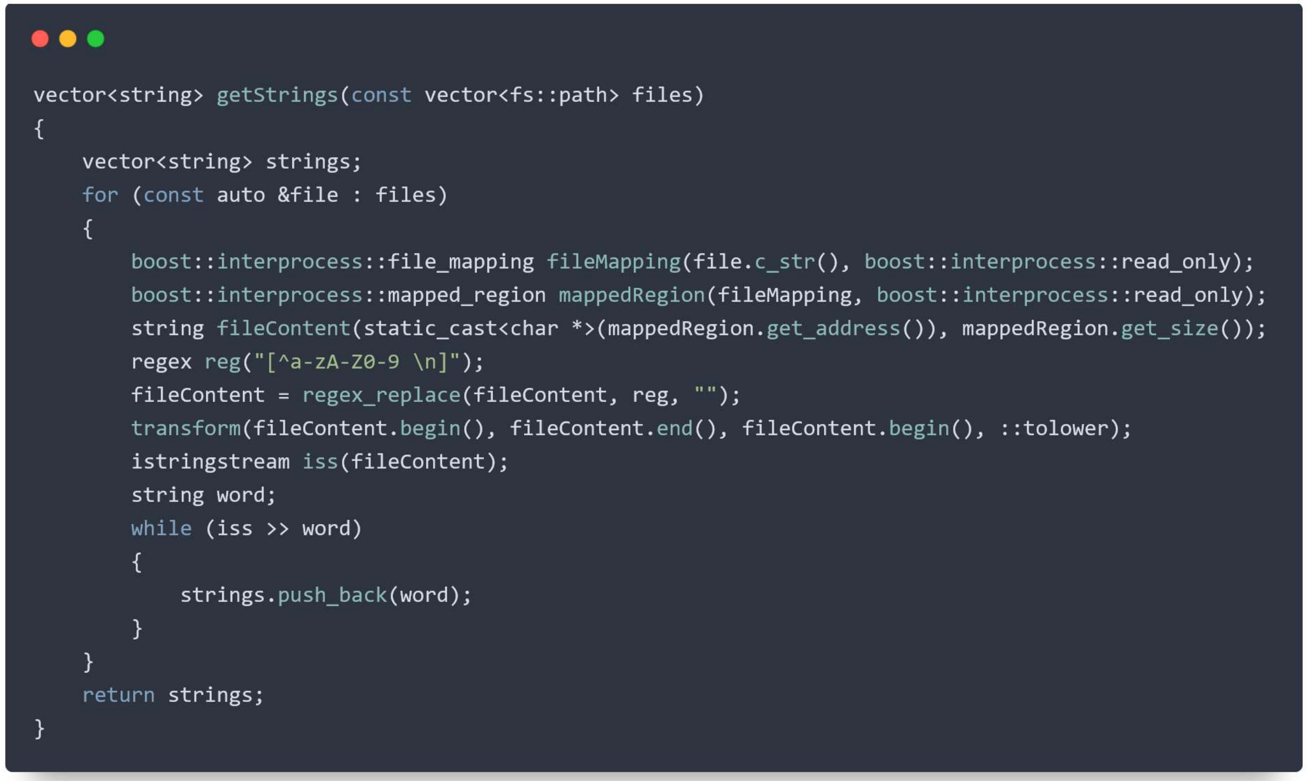
Jakob Friedl if20b089 – Philipp Haider if20b097

reading files



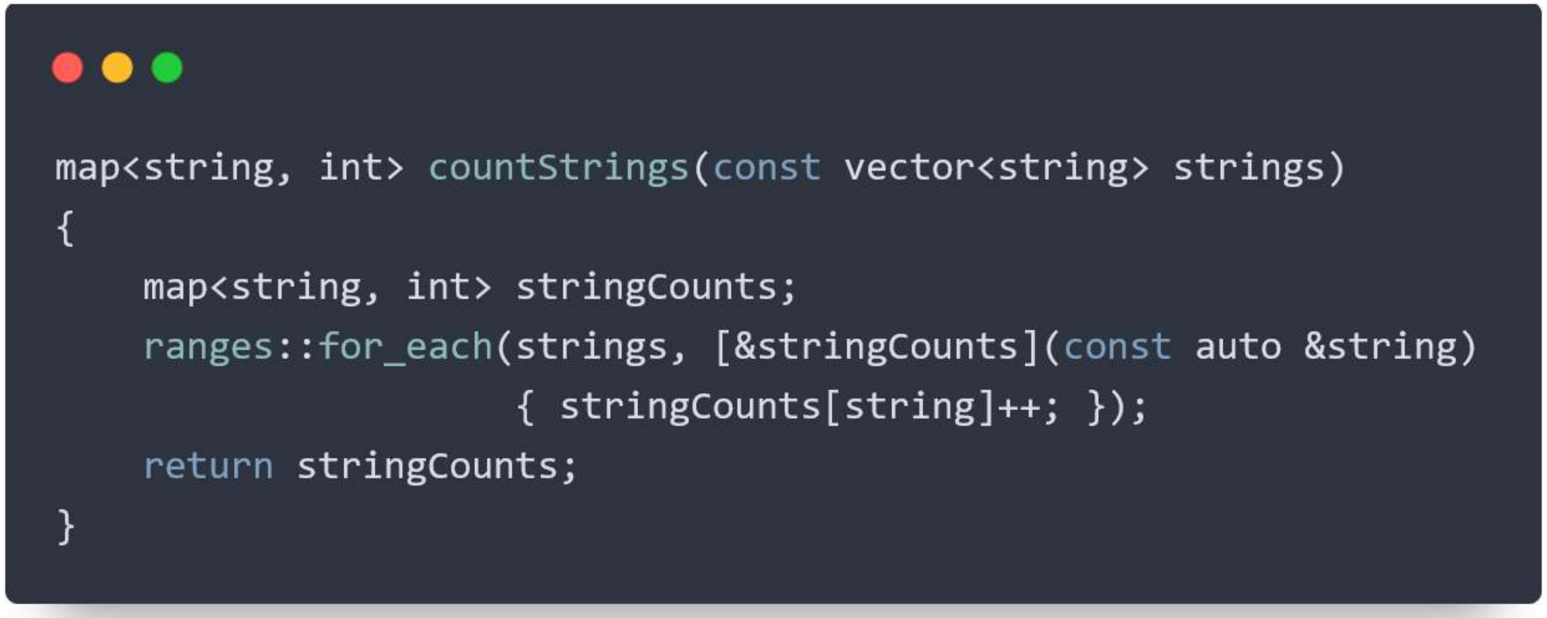
```cpp
vector<fs::path> getAllFiles(const fs::path path, const string extension)
{
    vector<fs::path> files;
    ranges::copy_if(fs::recursive_directory_iterator(path), back_inserter(files), [&extension](const auto &entry)
                    { return entry.is_regular_file() && entry.path().extension() == extension; });
    return files;
}
```

getting strings



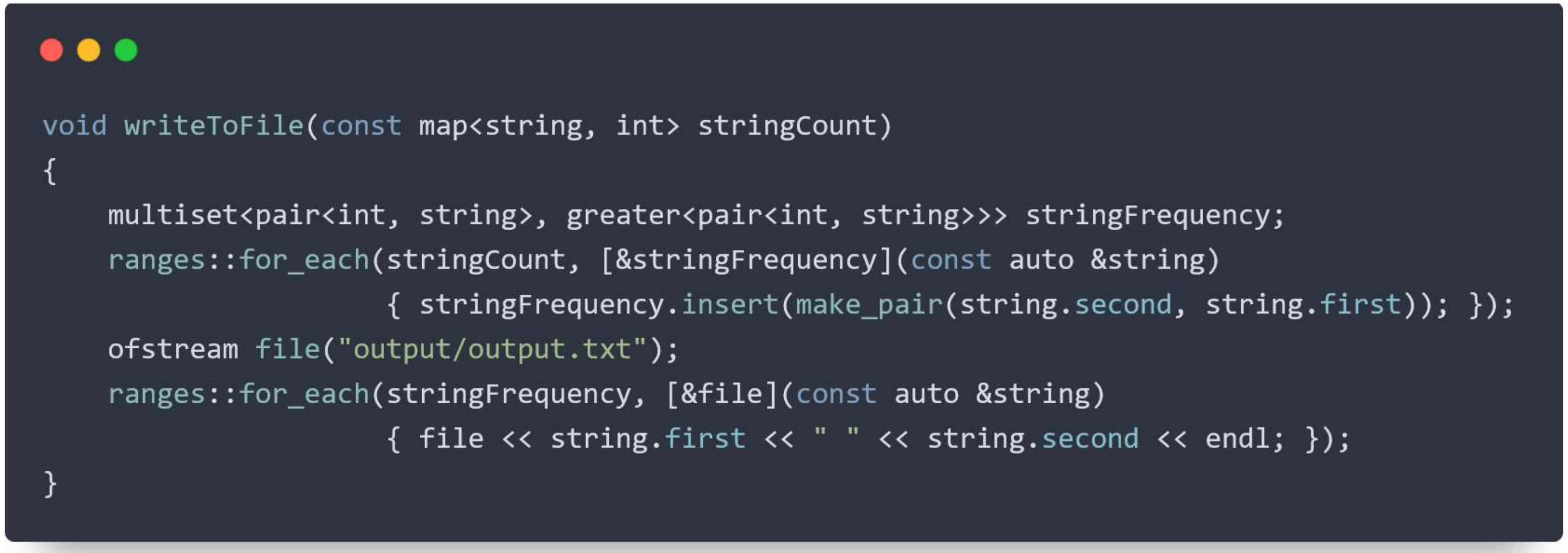
```cpp
vector<string> getStrings(const vector<fs::path> files)
{
    vector<string> strings;
    for (const auto &file : files)
    {
        boost::interprocess::file_mapping fileMapping(file.c_str(), boost::interprocess::read_only);
        boost::interprocess::mapped_region mappedRegion(fileMapping, boost::interprocess::read_only);
        string fileContent(static_cast<char *>(mappedRegion.get_address()), mappedRegion.get_size());
        regex reg("[^a-zA-Z0-9 \n]");
        fileContent = regex_replace(fileContent, reg, "");
        transform(fileContent.begin(), fileContent.end(), fileContent.begin(), ::tolower);
        istringstream iss(fileContent);
        string word;
        while (iss >> word)
        {
            strings.push_back(word);
        }
    }
    return strings;
}
```

counting strings



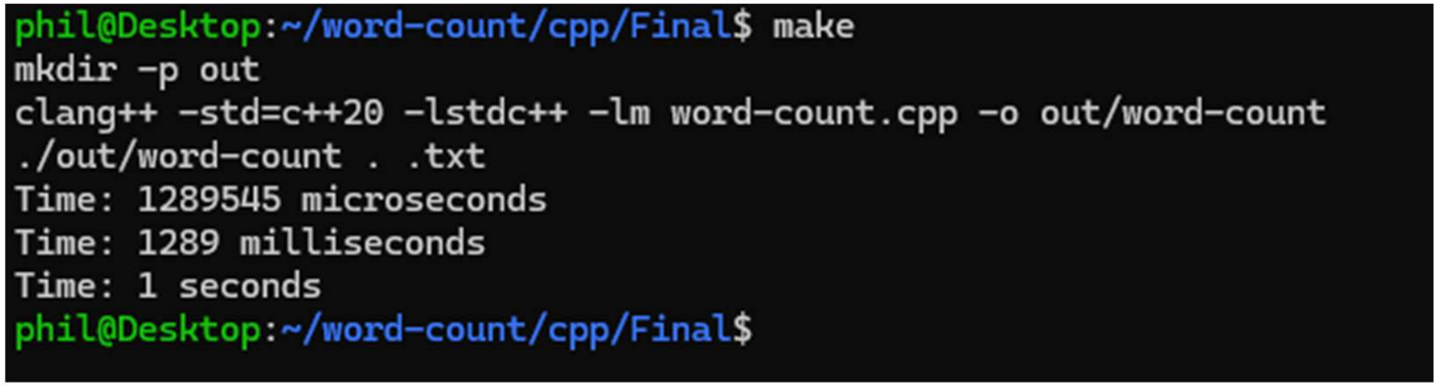
```
map<string, int> countStrings(const vector<string> strings)
{
    map<string, int> stringCounts;
    ranges::for_each(strings, [&stringCounts](const auto &string)
                     { stringCounts[string]++; });
    return stringCounts;
}
```

writing to output-file

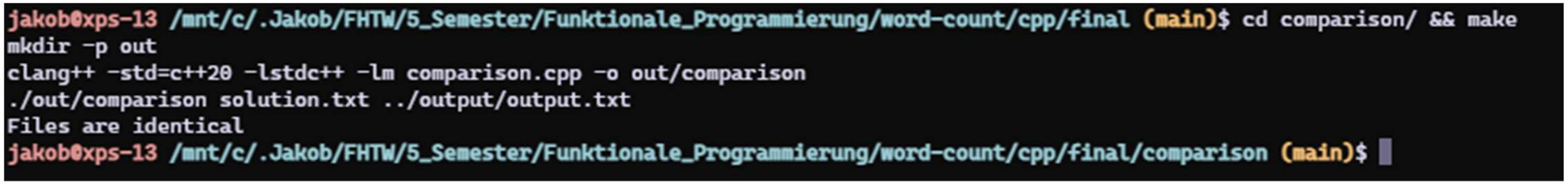


```cpp
void writeToFile(const map<string, int> stringCount)
{
    multiset<pair<int, string>, greater<pair<int, string>>> stringFrequency;
    ranges::for_each(stringCount, [&stringFrequency](const auto &string)
                     { stringFrequency.insert(make_pair(string.second, string.first)); });
    ofstream file("output/output.txt");
    ranges::for_each(stringFrequency, [&file](const auto &string)
                     { file << string.first << " " << string.second << endl; });
}
```

runtime analysis & tests



```
phil@Desktop:~/word-count/cpp/Final$ make
mkdir -p out
clang++ -std=c++20 -lstdc++ -lm word-count.cpp -o out/word-count
./out/word-count . .txt
Time: 1289545 microseconds
Time: 1289 milliseconds
Time: 1 seconds
phil@Desktop:~/word-count/cpp/Final$
```



```
jakob@xps-13 /mnt/c/.Jakob/FHTW/5_Semester/Funktionale_Programmierung/word-count/cpp/final (main)$ cd comparison/ && make
mkdir -p out
clang++ -std=c++20 -lstdc++ -lm comparison.cpp -o out/comparison
./out/comparison solution.txt ../output/output.txt
Files are identical
jakob@xps-13 /mnt/c/.Jakob/FHTW/5_Semester/Funktionale_Programmierung/word-count/cpp/final/comparison (main)$
```

functional concepts used

- Lambdas
- C++ Ranges
- Pure functions
- Immutability

lessons learned

- Functional programming concepts in C++
  - Lambdas, C++ Ranges
- Memory checks with valgrind
- Time optimization
  - libboost-Library
  - use of performance-optimized methods

failed/other approaches

- Runtime optimization by multithreading
- Functional word count in other programming languages
  - Python
  - Nim