Genealogical Tree

0.0.1_ddc4b71

Generated by Doxygen 1.8.9.1

Tue Jun 30 2015 00:49:23

Contents

Chapter 1

Genealogical Tree

Summary

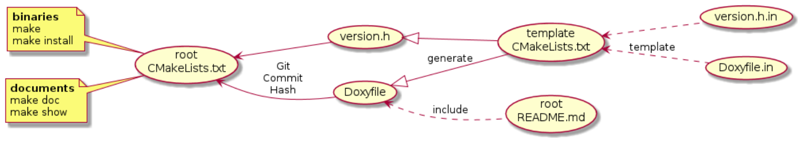
Program should be able to **find all the descendant with name Bob for all the ascendants with name Will on any level of ancestry**. In order to present the capabilities of your app:

- implement the application to optimize the initialization time
- application should have built in data about genealogical tree of people living in particular country
- please generate a representative data that has sample people an relationships between them. Use all varieties of names (can be also generated) but also put two test names (Bob and Will) and connect them in different relationships.
- the application should posses tests that are checking possible edge cases and ensure the stability of the application.
- the designed data structure should ensure optimized search time on following fields: name, last name, date of birth and location.

Generate binaries & documentation

Usual commands:

```
mkdir build
cd build
cmake ..
make
make install
make doc
```



Note: If you happen to work with *OSX* and `Homebrew`, don't forget to invoke *cmake* pointing to the **GNU** compiler:

```
cmake -DCMAKE_CXX_COMPILER=g++-5 ..
```

Note: If you happen to work with *Windows* and `Git`/`MinGW`, don't forget to invoke *cmake* pointing to the **GNU** generator:

```
cmake -G "MSYS Makefiles" ..
```

Development details

In order to generate binaries & documentation, the following versions were used:

For code

∗Linux∗ (Xubuntu 15.04)

- **cmake** *3.1.3*
- **gcc** *4.9.2*
- **boost** *1.55*

∗OSX∗ (Yosemite 10.10.3)

- **cmake** *3.2.2*
- **gcc** *5.1*
- **boost** *1.58*

Note: If you happen to work with *OSX* and `Homebrew`, don't forget to compile **boost** with the previous **gcc** compiler, not with the default *clang* one:

```
brew install gcc
brew install boost --cc=gcc-5
```

∗Windows∗ (Win7 x64)

- **cmake** *3.3.0*
- **gcc** *5.1*
- **boost** *1.58*

For documentation

∗Linux∗

- **doxygen** *1.8.9.1*
- **latex/pdfTeX** *2.6-1.40.15*
- **graphviz/dot** *2.38.0*
- **java/plantuml** *1.8.0_45/8026*

∗OSX∗

- **doxygen** *1.8.9.1*
- **latex/pdfTeX** *2.6-1.40.15*
- **graphviz/dot** *2.38.0*
- **java/plantuml** *1.8.0_40/8026*

∗Windows∗

- **doxygen** *1.8.9.1*
- **latex/pdfTeX** *2.9.5496-1.40.15*
- **graphviz/dot** *2.38.0*
- **java/plantuml** *1.8.0_45/8026*

Note: Don't forget configure *Doxyfile* and *CMakeLists.txt* to use **README.md** as *Main Page* for **latex** documentation.

For IDE

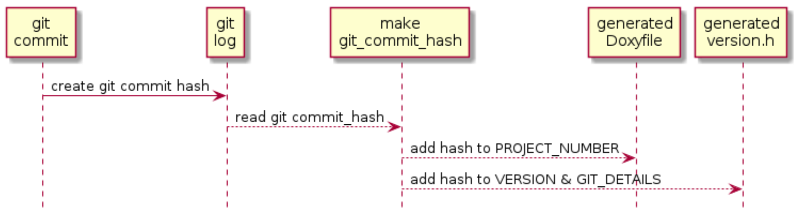
To use **NetBeans** don't forget to configure a *cmake* project with *custom* **build** folder. Add at that moment any extra customization in the command line used by *cmake* instruction. For example:

- -DCMAKE_CXX_COMPILER=g++-5 for **OSX**
- -G "MSYS Makefiles" for **Windows**

Note: If you happen to use *jVi* plugin on *OSX*, don't forget to use ∗∗-lc∗∗ instead of just ∗∗-c∗∗ for its ∗/bin/bash∗ flag.

GIT Commit Hash

In order to add the specific **git commit hash** into code & documentation, *templates* are defined in the *template* folder for **Doxyfile** & **version.h** files.



Chapter 2

Sources

Source folder for headears & code files.

Generate Files

version.h is generated with *GIT* information

Chapter 3

Tests

Future folder with *boost* test cases

Chapter 4

File Index

4.1 File List

Here is a list of all documented files with brief descriptions:

Chapter 5

File Documentation

5.1 src/main.cpp File Reference

```
#include <iostream>
#include <utility>
#include <algorithm>
#include <boost/graph/graph_traits.hpp>
#include <boost/graph/adjacency_list.hpp>
#include <boost/graph/dijkstra_shortest_paths.hpp>
#include "version.h"
```

Functions

- int main (int argc, char **argv)

  Main function.

5.1.1 Function Documentation

5.1.1.1 int main (int *argc*, char ** *argv*)

Main function.

Parameters

argc	An integer argument count of the command line arguments
argv	An argument vector of the command line arguments

Returns

an integer 0 upon exit success

5.2 src/version.h File Reference

Macros

- #define **DEFINE_VERSION_FIRST** "0"
- #define **DEFINE_VERSION_MIDDLE** "0"
- #define **DEFINE_VERSION_LAST** "1"

- #define **DEFINE_GIT_DETAILS** "ddc4b71 (HEAD, develop) Local CMake script to control git commit hash information and Markdown documentation at version.h"
- #define **DEFINE_GIT_COMMIT_HASH** "ddc4b71"
- #define **DEFINE_VERSION**

Variables

- static const char ∗ **VERSION** = "VERSION = " DEFINE_VERSION
- static const char ∗ **GIT_DETAILS** = "GIT_DETAILS = " DEFINE_GIT_DETAILS

5.2.1 Detailed Description

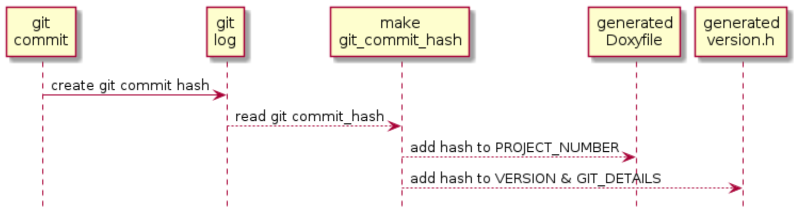
This metadata information might be located through **strings** command

- Linux/Solaris/Mac:

```
strings <binary> | grep VERSION
strings <binary> | grep GIT_DETAILS
```

- Windows (MinGW):

```
strings <binary> | findstr VERSION
strings <binary> | findstr GIT_DETAILS
```



5.2.2 Macro Definition Documentation

5.2.2.1 #define DEFINE_VERSION

Value:

```
DEFINE_VERSION_FIRST "." \
        DEFINE_VERSION_MIDDLE "." DEFINE_VERSION_LAST \
    "_" DEFINE_GIT_COMMIT_HASH
```

Index