Cpp concept project

Generated by Doxygen 1.8.20

Chapter 1

C++ concepts project

1.1 Idea

This project serves as sample/concept project for further projects :thumbsup:

1.2 Related documents

- `Notes`
- Markdown cheatsheet
- Project structure
- Unit testing

1.3 Structure

1.3.1 Folders

- **bin**: output executables go here (for the app, tests and spikes)
- **build**: containing all the object files (removed by clean)
- **doc**: documentation files
- **ideas**: smaller classes or files to test technologies or ideas
- **include**: all project header files, all necessary third-party header files (which are not in /usr/local/include)
- **lib**: any library that get compiled by the project, third party or any needed in development
- **resources**: resources
- **src**: the application and application's source files
- **test**: all test code files

1.4 Content (Concepts)

1.4.1 Programming concepts

- Classes
  - Inheritance
- Templates
- ...

1.4.2 Documentation

The documentation is intrinsically implemented using `doxygen`. In order to do that:

- specify path to doxygen binary in the Makefile
- execute *make doc*

The *README.md* file is used for the Mainpage of the documentation. Set the settings for doxygen in *doc/Doxyfile*.

1.4.3 Makefile

Following targets are implemented:

- **all** default make
- **remake**
- **clean**
- **cleaner**
- **resources**
- **sources**
- **directories**
- **ideas**
- **tester**
- **doc**

Chapter 2

Markdown cheatsheet

Short reference sheet for Markdown. Be aware that some things may not work properly in dependence of the used Markdown flavor.

2.1 Header 1

2.1.1 Header 2

2.1.1.1 Header 3

2.1.1.1.1 Header 4

Header 5

2.2 Emphasis

Emphasis, aka italics, with *asterisks* or *underscores*.

Strong emphasis, aka bold, with **asterisks** or **underscores**.

Combined emphasis with **asterisks and *underscores***.

Strikethrough uses two tildes. ~~Scratch this.~~

2.3 Lists

1. First ordered list item
2. Another item
   - Unordered sub-list.

1. Actual numbers don't matter, just that it's a number
   (a) Ordered sub-list
2. And another item.

   You can have properly indented paragraphs within list items. Notice the blank line above, and the leading spaces (at least one, but we'll use three here to also align the raw Markdown).

   To have a line break without a paragraph, you will need to use two trailing spaces. Note that this line is separate, but within the same paragraph. (This is contrary to the typical GFM line break behaviour, where trailing spaces are not required.)

- Unordered list can use asterisks
- Or minuses
- Or pluses

2.4 Links

I'm an inline-style link

I'm an inline-style link with title

I'm a reference-style link

You can use numbers for reference-style link definitions

Or leave it empty and use the link text itself.

URLs and URLs in angle brackets will automatically get turned into links. http://www.example.com or http://www.example.com and sometimes example.com (but not on Github, for example).

Some text to show that the reference links can follow later.

2.5 Images

Here's our logo (hover to see the title text):

Inline-style:

Reference-style:

2.6 Code and Syntax Highlighting

Inline `code` has `back-ticks around` it.

```
var s = "JavaScript syntax highlighting";
alert(s);
s = "Python syntax highlighting"
print(s)
No language indicated, so no syntax highlighting.
But let’s throw in a <b>tag</b>.
```

2.7 Tables

Colons can be used to align columns.

Tables	Are	Cool
col 3 is	right-aligned	$1600
col 2 is	centered	$12
zebra stripes	are neat	$1

There must be at least 3 dashes separating each header cell. The outer pipes (|) are optional, and you don't need to make the raw Markdown line up prettily. You can also use inline Markdown.

Markdown	Less	Pretty
Still	`renders`	**nicely**
1	2	3

2.8 Blockquotes

> Blockquotes are very handy in email to emulate reply text. This line is part of the same quote.

Quote break.

> This is a very long line that will still be quoted properly when it wraps. Oh boy let's keep writing to make sure this is long enough to actually wrap for everyone. Oh, you can *put* **Markdown** into a blockquote.

2.9 Inline HTML

You can also use raw HTML in your Markdown, and it'll mostly work pretty well.

Definition list Is something people use sometimes.

Markdown in HTML Does *not* work **very** well. Use HTML *tags*.

2.10 Horizontal

Three or more...

Hyphens

Asterisks

Underscores

2.11 YouTube Videos

They can't be added directly but you can add an image with a link to the video like this:

Or, in pure Markdown, but losing the image sizing and border:

Referencing a bug by #bugID in your git commit links it to the slip. For example #1.

Chapter 3

Project structure

3.1 Folders

- **bin**: output executables go here (for the app, tests and spikes)
- **build**: containing all the object files (removed by clean)
- **doc**: documentation files
- **include**: all project header files, all necessary third-party header files (which are not in /usr/local/include)
- **lib**: any library that get compiled by the project, third party or any needed in development
- **spike**: smaller classes or files to test technologies or ideas
- **src**: the application and application's source files
- **test**: all test code files

3.2 Files

- **Makefile**: Makefile
- **README.md**: Readme file in markdown syntax

Chapter 4

Unit-Tests

4.1 Integrated in CLion

4.1.1 Google Test

See Googletest - google Testing and Mocking Framework `Google test` on Github.

4.1.2 Catch

See `Catch Org` and `Catch2` for a modern, C++ native, header only test framework for unit-tests, TDD and BDD.

4.1.3 Boost.Test

See the `Boost.test` for the C++ Boost.Test library, providing both an easy to use and flexible set of interfaces for writing test programs, organizing tests into simple test cases and test suites, and controlling their runtime execution.

4.1.4 Doctest

`Doctest` is a new C++ testing framework but is by far the fastest both in compile times (by orders of magnitude) and runtime compared to other feature-rich alternatives. It brings the ability of compiled languages such as D / Rust / Nim to have tests written directly in the production code thanks to a fast, transparent and flexible test runner with a clean interface.

Chapter 5

Class Index

5.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

Chapter 6

File Index

6.1 File List

Here is a list of all files with brief descriptions:

Chapter 7

Class Documentation

7.1 ConceptClass Class Reference

```
#include "ConceptClass.h"
```

Public Member Functions

- ConceptClass (int a, int b)

Public Attributes

- int member_a
- int member_b

7.1.1 Detailed Description

Definition at line 12 of file ConceptClass.h.

7.1.2 Constructor & Destructor Documentation

7.1.2.1 ConceptClass()

```
ConceptClass::ConceptClass (
              int a,
              int b )
```

Constructor
Detailed description for constructor.

Parameters

a	
b	

Definition at line 3 of file ConceptClass.cpp.

```
00003                                              {
00004     member_a = a;
00005     member_b = b;
00006 }
```

7.1.3 Member Data Documentation

7.1.3.1 member_a

```
int ConceptClass::member_a
```

Parameters

member	a

Definition at line 22 of file ConceptClass.h.

7.1.3.2 member_b

```
int ConceptClass::member_b
```

Parameters

member	b

Definition at line 24 of file ConceptClass.h.
The documentation for this class was generated from the following files:

- include/ConceptClass.h
- src/ConceptClass.cpp

Chapter 8

File Documentation

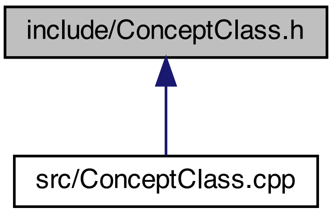
8.1 documents/Markdown.md File Reference

8.2 documents/structure.md File Reference

8.3 documents/Unit-Tests.md File Reference

8.4 include/ConceptClass.h File Reference

This graph shows which files directly or indirectly include this file:



Classes

- class ConceptClass

8.5 ConceptClass.h

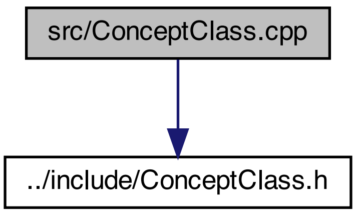
```
#ifndef CPP_CONCEPTS_PROJECT_CONCEPTCLASS_H
#define CPP_CONCEPTS_PROJECT_CONCEPTCLASS_H

class ConceptClass {
public:
    ConceptClass(int a,int b);

    int member_a;
    int member_b;
};


#endif //CPP_CONCEPTS_PROJECT_CONCEPTCLASS_H
```

8.6 README.md File Reference

8.7 src/ConceptClass.cpp File Reference

```
#include "../include/ConceptClass.h"
```

Include dependency graph for ConceptClass.cpp:



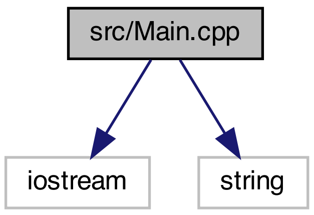
8.8 ConceptClass.cpp

```
#include "../include/ConceptClass.h"

ConceptClass::ConceptClass(int a, int b) {
    member_a = a;
    member_b = b;
}
```

8.9 src/Main.cpp File Reference

```
#include <iostream>
#include <string>
```

Include dependency graph for Main.cpp:



Functions

- int main ()

8.9.1 Function Documentation

8.9.1.1 main()

`int main ( )`

Definition at line 4 of file Main.cpp.

```
00004             {
00005 
00006     printf("Hello World!\n");
00007 
00008     return 0;
00009 }
```

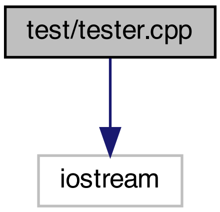
8.10 Main.cpp

```
00001 #include <iostream>
00002 #include <string>
00003 
00004 int main() {
00005 
00006     printf("Hello World!\n");
00007 
00008     return 0;
00009 }
00010 
```

8.11 test/tester.cpp File Reference

`#include <iostream>`

Include dependency graph for tester.cpp:



Functions

- int main ()

8.11.1 Function Documentation

8.11.1.1 main()

`int main ( )`

Definition at line 3 of file tester.cpp.

```
00003             {
00004 
00005     std::cout « "This is a tester file" « std::endl;
00006     return 0;
00007 
00008 }
```

8.12 tester.cpp

```
#include <iostream>

int main() {

    std::cout « "This is a tester file" « std::endl;
    return 0;

}

```