

Gigabrot_C++

Generated by Doxygen 1.9.2

1 Gigabrot	1
1.1 About The Project	1
1.1.1 Note:	2
1.1.1.1 Built With:	2
1.2 Getting Started	2
1.2.1 Prerequisites	2
1.2.2 Cloning	2
1.3 Usage	2
1.4 Contact	3
1.4.1 Demonstrated C++ Fundamentals	3
1.4.1.1 For Reference	3
2 Hierarchical Index	5
2.1 Class Hierarchy	5
3 Class Index	7
3.1 Class List	7
4 File Index	9
4.1 File List	9
5 Class Documentation	11
5.1 Colorization Class Reference	11
5.1.1 Detailed Description	12
5.1.2 Constructor & Destructor Documentation	12
5.1.2.1 ~Colorization()	13
5.1.2.2 Colorization()	13
5.1.3 Member Function Documentation	13
5.1.3.1 get_max_color_value()	13
5.1.3.2 get_min_color_value()	13
5.1.3.3 get_type()	13
5.1.4 Member Data Documentation	14
5.1.4.1 maxColorValue	14
5.1.4.2 minColorValue	14
5.1.4.3 type	14
5.2 InsideColor Class Reference	15
5.2.1 Detailed Description	17
5.2.2 Constructor & Destructor Documentation	17
5.2.2.1 InsideColor()	17
5.2.2.2 ~InsideColor()	17
5.2.3 Member Function Documentation	17
5.2.3.1 calculate_b()	17
5.2.3.2 calculate_bw()	18
5.2.3.3 calculate_g()	18

5.2.3.4 calculate_r()	18
5.3 LineColor Class Reference	19
5.3.1 Detailed Description	21
5.3.2 Constructor & Destructor Documentation	21
5.3.2.1 LineColor()	21
5.3.2.2 ~LineColor()	21
5.3.3 Member Function Documentation	21
5.3.3.1 calculate_b()	21
5.3.3.2 calculate_bw()	22
5.3.3.3 calculate_g()	22
5.3.3.4 calculate_r()	22
5.4 Mandelbrot Class Reference	23
5.4.1 Detailed Description	25
5.4.2 Constructor & Destructor Documentation	25
5.4.2.1 Mandelbrot() [1/3]	25
5.4.2.2 Mandelbrot() [2/3]	25
5.4.2.3 Mandelbrot() [3/3]	26
5.4.2.4 ~Mandelbrot()	26
5.4.3 Member Function Documentation	27
5.4.3.1 average()	27
5.4.3.2 colorize_bw()	28
5.4.3.3 current_pixel()	28
5.4.3.4 describe_border()	29
5.4.3.5 get_c()	30
5.4.3.6 get_t()	30
5.4.3.7 in_border()	31
5.4.3.8 in_set()	31
5.4.3.9 interpolate()	32
5.4.3.10 iterate()	32
5.4.3.11 reset()	33
5.4.3.12 set_border()	34
5.4.3.13 set_image()	34
5.4.3.14 set_iSkip()	34
5.4.3.15 set_plane()	34
5.4.3.16 set_stripe_density()	35
5.4.3.17 shape_check()	35
5.4.4 Friends And Related Function Documentation	35
5.4.4.1 operator<<	35
5.4.5 Member Data Documentation	36
5.4.5.1 a	36
5.4.5.2 bulb	36
5.4.5.3 c	36

5.4.5.4 cardioid	36
5.4.5.5 cxMax	36
5.4.5.6 cxMin	37
5.4.5.7 cyMax	37
5.4.5.8 cyMin	37
5.4.5.9 d	37
5.4.5.10 dC	37
5.4.5.11 de	37
5.4.5.12 escapeRadius	38
5.4.5.13 height	38
5.4.5.14 iSkip	38
5.4.5.15 iter	38
5.4.5.16 iterMax	38
5.4.5.17 pixHeight	38
5.4.5.18 pixWidth	39
5.4.5.19 prevA	39
5.4.5.20 pX	39
5.4.5.21 pY	39
5.4.5.22 q	39
5.4.5.23 r	39
5.4.5.24 shade	40
5.4.5.25 stripeDensity	40
5.4.5.26 thin	40
5.4.5.27 width	40
5.4.5.28 z	40
5.5 Neumorphic Class Reference	41
5.5.1 Detailed Description	43
5.5.2 Constructor & Destructor Documentation	43
5.5.2.1 Neumorphic()	43
5.5.3 Member Function Documentation	43
5.5.3.1 calculate()	43
5.5.3.2 get_angle()	44
5.5.3.3 get_heightFactor()	44
5.5.3.4 get_reflection()	45
5.5.4 Member Data Documentation	45
5.5.4.1 angle	45
5.5.4.2 dC	45
5.5.4.3 heightFactor	45
5.5.4.4 reflection	45
5.5.4.5 u	46
5.5.4.6 v	46
5.5.4.7 z	46

5.6 NormalMap Class Reference	46
5.6.1 Detailed Description	48
5.6.2 Constructor & Destructor Documentation	48
5.6.2.1 ~NormalMap()	49
5.6.2.2 NormalMap()	49
5.6.3 Member Function Documentation	49
5.6.3.1 calculate()	49
5.6.3.2 dot_product()	49
5.6.3.3 get_max_val()	50
5.6.3.4 get_min_val()	50
5.6.3.5 get_type()	50
5.6.4 Member Data Documentation	50
5.6.4.1 maxMapVal	50
5.6.4.2 minMapVal	51
5.6.4.3 type	51
5.7 PPM Class Reference	51
5.7.1 Detailed Description	53
5.7.2 Constructor & Destructor Documentation	53
5.7.2.1 PPM() [1/3]	53
5.7.2.2 PPM() [2/3]	53
5.7.2.3 PPM() [3/3]	54
5.7.3 Member Function Documentation	54
5.7.3.1 close()	54
5.7.3.2 init_stream()	55
5.7.3.3 operator=()	55
5.7.3.4 set_comment()	56
5.7.3.5 set_height()	56
5.7.3.6 set_outputDirectory()	56
5.7.3.7 set_width()	57
5.7.3.8 write_header()	57
5.7.3.9 write_row()	58
5.7.4 Member Data Documentation	58
5.7.4.1 comment	58
5.7.4.2 fileName	58
5.7.4.3 header	59
5.7.4.4 height	59
5.7.4.5 image	59
5.7.4.6 magic	59
5.7.4.7 outputDirectory	59
5.7.4.8 pixMaxVal	59
5.7.4.9 subPixel	60
5.7.4.10 width	60

5.8 Shading Class Reference	60
5.8.1 Detailed Description	62
5.8.2 Constructor & Destructor Documentation	62
5.8.2.1 Shading()	62
5.8.2.2 ~Shading()	62
5.8.3 Member Function Documentation	62
5.8.3.1 calculate_b()	62
5.8.3.2 calculate_bw()	62
5.8.3.3 calculate_g()	63
5.8.3.4 calculate_r()	63
5.8.3.5 get_max_color_value()	63
5.8.3.6 get_min_color_value()	63
5.8.4 Member Data Documentation	63
5.8.4.1 type	63
5.9 Striping Class Reference	64
5.9.1 Detailed Description	66
5.9.2 Constructor & Destructor Documentation	66
5.9.2.1 Striping()	66
5.9.2.2 ~Striping()	66
5.9.3 Member Function Documentation	66
5.9.3.1 calculate_b()	66
5.9.3.2 calculate_bw()	67
5.9.3.3 calculate_g()	67
5.9.3.4 calculate_r()	67
5.9.3.5 get_average()	68
5.9.4 Member Data Documentation	68
5.9.4.1 average	68
5.9.4.2 reflection	68
5.10 ThreadPool Class Reference	68
5.10.1 Detailed Description	70
5.10.2 Member Typedef Documentation	70
5.10.2.1 process	70
5.10.2.2 queue	70
5.10.2.3 queueVec	70
5.10.2.4 Threads	70
5.10.3 Constructor & Destructor Documentation	70
5.10.3.1 ThreadPool()	71
5.10.3.2 ~ThreadPool()	71
5.10.4 Member Function Documentation	72
5.10.4.1 enqueue_task()	72
5.10.4.2 enqueue_work()	72
5.10.5 Member Data Documentation	73

5.10.5.1 count	73
5.10.5.2 countMult	73
5.10.5.3 index	73
5.10.5.4 queues	73
5.10.5.5 threads	74
5.11 UnboundedQueue< T > Class Template Reference	74
5.11.1 Detailed Description	75
5.11.2 Member Typedef Documentation	75
5.11.2.1 queue_t	75
5.11.3 Constructor & Destructor Documentation	76
5.11.3.1 UnboundedQueue()	76
5.11.3.2 ~UnboundedQueue()	76
5.11.4 Member Function Documentation	76
5.11.4.1 block()	76
5.11.4.2 blocking()	77
5.11.4.3 emplace()	77
5.11.4.4 empty()	77
5.11.4.5 pop()	78
5.11.4.6 push() [1/2]	78
5.11.4.7 push() [2/2]	78
5.11.4.8 size()	79
5.11.4.9 try_pop()	79
5.11.4.10 try_push() [1/2]	79
5.11.4.11 try_push() [2/2]	80
5.11.4.12 unblock()	80
5.11.5 Member Data Documentation	80
5.11.5.1 condition	80
5.11.5.2 is_block	81
5.11.5.3 queue	81
5.11.5.4 queueLock	81
6 File Documentation	83
6.1 D:/Documents/Final/Mandelbrot_Cpp_Final/cmake-build-debug/CMakeFiles/3.21.1/CompilerIdC/↵ CMakeCCompilerId.c File Reference	83
6.1.1 Macro Definition Documentation	83
6.1.1.1 __has_include	84
6.1.1.2 ARCHITECTURE_ID	84
6.1.1.3 C_DIALECT	84
6.1.1.4 COMPILER_ID	84
6.1.1.5 DEC	84
6.1.1.6 HEX	85
6.1.1.7 PLATFORM_ID	85
6.1.1.8 STRINGIFY	85

6.1.1.9 STRINGIFY_HELPER	85
6.1.2 Function Documentation	85
6.1.2.1 main()	86
6.1.3 Variable Documentation	86
6.1.3.1 info_arch	86
6.1.3.2 info_compiler	86
6.1.3.3 info_language_dialect_default	86
6.1.3.4 info_platform	87
6.2 CMakeCCompilerId.c	87
6.3 D:/Documents/Final/Mandelbrot_Cpp_Final/cmake-build-debug/CMakeFiles/3.21.1/CompilerIdCXX/CMakeCXXCompilerId.cpp File Reference	96
6.3.1 Macro Definition Documentation	97
6.3.1.1 __has_include	97
6.3.1.2 ARCHITECTURE_ID	97
6.3.1.3 COMPILER_ID	97
6.3.1.4 CXX_STD	97
6.3.1.5 DEC	98
6.3.1.6 HEX	98
6.3.1.7 PLATFORM_ID	98
6.3.1.8 STRINGIFY	98
6.3.1.9 STRINGIFY_HELPER	99
6.3.2 Function Documentation	99
6.3.2.1 main()	99
6.3.3 Variable Documentation	99
6.3.3.1 info_arch	99
6.3.3.2 info_compiler	100
6.3.3.3 info_language_dialect_default	100
6.3.3.4 info_platform	100
6.4 CMakeCXXCompilerId.cpp	100
6.5 D:/Documents/Final/Mandelbrot_Cpp_Final/cmake-build-debug/CMakeFiles/FindOpenMP/OpenMPCheckVersion.c File Reference	109
6.5.1 Function Documentation	110
6.5.1.1 main()	110
6.5.2 Variable Documentation	110
6.5.2.1 ompver_str	111
6.6 OpenMPCheckVersion.c	111
6.7 D:/Documents/Final/Mandelbrot_Cpp_Final/cmake-build-debug/CMakeFiles/FindOpenMP/OpenMPCheckVersion.cpp File Reference	111
6.7.1 Function Documentation	112
6.7.1.1 main()	112
6.7.2 Variable Documentation	112
6.7.2.1 ompver_str	112
6.8 OpenMPCheckVersion.cpp	112

6.9	D:/Documents/Final/Mandelbrot_Cpp_Final/cmake-build-debug/CMakeFiles/FindOpenMP/Open↔ MPTryFlag.c File Reference	113
6.9.1	Function Documentation	113
6.9.1.1	main()	113
6.10	OpenMPTryFlag.c	113
6.11	D:/Documents/Final/Mandelbrot_Cpp_Final/cmake-build-debug/CMakeFiles/FindOpenMP/Open↔ MPTryFlag.cpp File Reference	114
6.11.1	Function Documentation	114
6.11.1.1	main()	114
6.12	OpenMPTryFlag.cpp	114
6.13	D:/Documents/Final/Mandelbrot_Cpp_Final/README.md File Reference	115
6.14	D:/Documents/Final/Mandelbrot_Cpp_Final/src/Colorization.cpp File Reference	115
6.15	Colorization.cpp	115
6.16	D:/Documents/Final/Mandelbrot_Cpp_Final/src/Colorization.h File Reference	116
6.17	Colorization.h	116
6.18	D:/Documents/Final/Mandelbrot_Cpp_Final/src/InsideColor.cpp File Reference	117
6.19	InsideColor.cpp	117
6.20	D:/Documents/Final/Mandelbrot_Cpp_Final/src/InsideColor.h File Reference	118
6.21	InsideColor.h	119
6.22	D:/Documents/Final/Mandelbrot_Cpp_Final/src/LineColor.cpp File Reference	120
6.23	LineColor.cpp	120
6.24	D:/Documents/Final/Mandelbrot_Cpp_Final/src/LineColor.h File Reference	121
6.25	LineColor.h	122
6.26	D:/Documents/Final/Mandelbrot_Cpp_Final/src/main.cpp File Reference	122
6.26.1	Function Documentation	123
6.26.1.1	main()	123
6.27	main.cpp	124
6.28	D:/Documents/Final/Mandelbrot_Cpp_Final/src/Mandelbrot.cpp File Reference	126
6.28.1	Function Documentation	126
6.28.1.1	operator<<()	126
6.29	Mandelbrot.cpp	127
6.30	D:/Documents/Final/Mandelbrot_Cpp_Final/src/Mandelbrot.h File Reference	130
6.30.1	Macro Definition Documentation	131
6.30.1.1	M_PI	131
6.31	Mandelbrot.h	131
6.32	D:/Documents/Final/Mandelbrot_Cpp_Final/src/Neumorphic.cpp File Reference	133
6.33	Neumorphic.cpp	133
6.34	D:/Documents/Final/Mandelbrot_Cpp_Final/src/Neumorphic.h File Reference	134
6.35	Neumorphic.h	135
6.36	D:/Documents/Final/Mandelbrot_Cpp_Final/src/NormalMap.cpp File Reference	136
6.37	NormalMap.cpp	136
6.38	D:/Documents/Final/Mandelbrot_Cpp_Final/src/NormalMap.h File Reference	136
6.39	NormalMap.h	137

6.40 D:/Documents/Final/Mandelbrot_Cpp_Final/src/PPM.cpp File Reference	138
6.41 PPM.cpp	138
6.42 D:/Documents/Final/Mandelbrot_Cpp_Final/src/PPM.h File Reference	139
6.43 PPM.h	140
6.44 D:/Documents/Final/Mandelbrot_Cpp_Final/src/Shading.cpp File Reference	141
6.45 Shading.cpp	142
6.46 D:/Documents/Final/Mandelbrot_Cpp_Final/src/Shading.h File Reference	143
6.47 Shading.h	143
6.48 D:/Documents/Final/Mandelbrot_Cpp_Final/src/Striping.cpp File Reference	144
6.49 Striping.cpp	145
6.50 D:/Documents/Final/Mandelbrot_Cpp_Final/src/Striping.h File Reference	145
6.51 Striping.h	146
6.52 D:/Documents/Final/Mandelbrot_Cpp_Final/src/ThreadPool.cpp File Reference	147
6.53 ThreadPool.cpp	147
6.54 D:/Documents/Final/Mandelbrot_Cpp_Final/src/ThreadPool.h File Reference	148
6.55 ThreadPool.h	149
6.56 D:/Documents/Final/Mandelbrot_Cpp_Final/src/ThreadPoolTest.cpp File Reference	150
6.56.1 Function Documentation	151
6.56.1.1 main()	151
6.57 ThreadPoolTest.cpp	152
6.58 D:/Documents/Final/Mandelbrot_Cpp_Final/src/UnboundedQueue.h File Reference	153
6.59 UnboundedQueue.h	154
Index	157

Chapter 1

Gigabrot

Gigabrot_C++

This is a C++ implementation of my original Gigabrot project, put into this repo for ease of submission. Its goal is to take in user-specified parameters and efficiently produce a [PPM](#) image render of the [Mandelbrot](#) set with stripe-average colorization and a pseudo-neumorphic normal mapping effect. This project also demonstrates various fundamental aspects of coding using the C++ language. Sample output image shown above.

[Jump to Usage»](#)

· [Report Bug](#) / [Request Feature](#) ·

1.1 About The Project

The [Mandelbrot set](#) is a set of complex numbers named for [Benoit B. Mandelbrot](#) for which the function $f_c(z) = z^2 + c$ does not exist when z is iterated from 0. The implications of this set of numbers range far and wide in the world of mathematics, and it is often a subject of interest in computing due to its many notable properties and ease of computing. It is also widely known for its aesthetic appeal as a fractal and the many coloring techniques that can be implemented.

The stripe average colorization method implemented here, described in Jussi Härkönen's "On fractal coloring techniques", is an extension of the Triangle Inequality Average method and highlights various features of the fractal. It is combined here with a normal vector map calculated to simulate the soft, shadowy nature of [modern neumorphic UI design](#).

A key aspect of the set's applications in computing is that the iteration is *embarrassingly parallel*, meaning it can be easily separated into parallel tasks. Thus, the creation of the thread pool in this project. This pool uses only C++ provide libraries and uses an unlocked work queue to allow for job-stealing, inspired by [Sean Parent's talk](#).

1.1.1 Note:

Parallelization of the main [Mandelbrot](#) code has not been implemented. The vector used to cache each row of the set to be written to the [PPM](#) file stream output throws a wrench in the "embarrassingly parallel" mix and I did not have time to fully debug the corrupted image output. While I removed that part of the code from my `main()`, I have included the test program I used to verify that the thread pool and its exception handling are in fact working.

1.1.1.1 Built With:

- CLion
- Cygwin

1.2 Getting Started

To get a local copy up and running follow these simple steps:

1.2.1 Prerequisites

- C++17
- CMake 3.21

To view the output `.ppm` files and convert to `.png`:

- [Netpbm Viewer](#)

1.2.2 Cloning

1. Clone the repo into the desired directory

```
git clone https://github.com/c-biancone/Mandelbrot_Cpp_Final.git
```
2. Open the directory from within CLion and build

1.3 Usage

1. In CLion, hit the play button to build and run the code. In the terminal, enter the image resolution when prompted. This value is automatically checked and copied so that the image is square for this implementation. Enter the desired filename when prompted.
2. Once the code has finished running, the output can be found within the `output` directory. A 1280 x 1280 image is generated in ~ 2.5 s on my laptop, and scales as expected according to image size.
3. Drag and drop this file into [Netpbm Viewer](#).
4. If this file is to your liking, download the `.png` file.

1.3.0.0.1 Test Thread Pool:

1. Find `CMakeLists.txt` and switch the comment under "executables" from `src/ThreadPoolTest.cpp` to `src/main.cpp`.
2. Build and run code as before. This main function will generate a random number of exceptions thrown from within the threads of the thread pool. Run repeatedly to get different output.

1.4 Contact

Chris Biancone - [email](#)

Project Link: <https://github.com/pinecone19/Gigabrot>

1.4.1 Demonstrated C++ Fundamentals

1.4.1.1 For Reference

- Class Inheritance
 - 3-level hierarchy
 - Is-a and Has-a relationship
 - Polymorphism
 - * [pure] virtual functions
 - * Abstract classes
 - * Downcasting
- Operator Overloading
 - Copy constructor
 - Assignment and stream insertion operators
- Data Structures
 - (Unbounded) queue
 - Vector
- Exception handling
 - Try/catch
 - Throw/rethrow
- C++ STL
- Custom Templated Classes
- File Processing
- String Stream Processing

Chapter 2

Hierarchical Index

2.1 Class Hierarchy

This inheritance list is sorted roughly, but not completely, alphabetically:

Colorization	11
Shading	60
InsideColor	15
LineColor	19
Striping	64
Mandelbrot	23
NormalMap	46
Neumorphic	41
PPM	51
ThreadPool	68
UnboundedQueue< T >	74

Chapter 3

Class Index

3.1 Class List

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

Colorization	11
InsideColor	15
LineColor	19
Mandelbrot	23
Neumorphic	41
NormalMap	46
PPM	51
Shading	60
Striping	64
ThreadPool	68
UnboundedQueue< T >	74

Chapter 4

File Index

4.1 File List

Here is a list of all files with brief descriptions:

D:/Documents/Final/Mandelbrot_Cpp_Final/cmake-build-debug/CMakeFiles/3.21.1/CompilerIdC/CMakeCCompilerId.c	83
D:/Documents/Final/Mandelbrot_Cpp_Final/cmake-build-debug/CMakeFiles/3.21.1/CompilerIdCXX/CMakeCXXCompilerId.cpp	96
D:/Documents/Final/Mandelbrot_Cpp_Final/cmake-build-debug/CMakeFiles/FindOpenMP/OpenMPCheckVersion.c	109
D:/Documents/Final/Mandelbrot_Cpp_Final/cmake-build-debug/CMakeFiles/FindOpenMP/OpenMPCheckVersion.cpp	111
D:/Documents/Final/Mandelbrot_Cpp_Final/cmake-build-debug/CMakeFiles/FindOpenMP/OpenMPTryFlag.c	113
D:/Documents/Final/Mandelbrot_Cpp_Final/cmake-build-debug/CMakeFiles/FindOpenMP/OpenMPTryFlag.cpp	114
D:/Documents/Final/Mandelbrot_Cpp_Final/src/Colorization.cpp	115
D:/Documents/Final/Mandelbrot_Cpp_Final/src/Colorization.h	116
D:/Documents/Final/Mandelbrot_Cpp_Final/src/InsideColor.cpp	117
D:/Documents/Final/Mandelbrot_Cpp_Final/src/InsideColor.h	118
D:/Documents/Final/Mandelbrot_Cpp_Final/src/LineColor.cpp	120
D:/Documents/Final/Mandelbrot_Cpp_Final/src/LineColor.h	121
D:/Documents/Final/Mandelbrot_Cpp_Final/src/main.cpp	122
D:/Documents/Final/Mandelbrot_Cpp_Final/src/Mandelbrot.cpp	126
D:/Documents/Final/Mandelbrot_Cpp_Final/src/Mandelbrot.h	130
D:/Documents/Final/Mandelbrot_Cpp_Final/src/Neumorphic.cpp	133
D:/Documents/Final/Mandelbrot_Cpp_Final/src/Neumorphic.h	134
D:/Documents/Final/Mandelbrot_Cpp_Final/src/NormalMap.cpp	136
D:/Documents/Final/Mandelbrot_Cpp_Final/src/NormalMap.h	136
D:/Documents/Final/Mandelbrot_Cpp_Final/src/PPM.cpp	138
D:/Documents/Final/Mandelbrot_Cpp_Final/src/PPM.h	139
D:/Documents/Final/Mandelbrot_Cpp_Final/src/Shading.cpp	141
D:/Documents/Final/Mandelbrot_Cpp_Final/src/Shading.h	143
D:/Documents/Final/Mandelbrot_Cpp_Final/src/Striping.cpp	144
D:/Documents/Final/Mandelbrot_Cpp_Final/src/Striping.h	145
D:/Documents/Final/Mandelbrot_Cpp_Final/src/ThreadPool.cpp	147
D:/Documents/Final/Mandelbrot_Cpp_Final/src/ThreadPool.h	148
D:/Documents/Final/Mandelbrot_Cpp_Final/src/ThreadPoolTest.cpp	150
D:/Documents/Final/Mandelbrot_Cpp_Final/src/UnboundedQueue.h	153

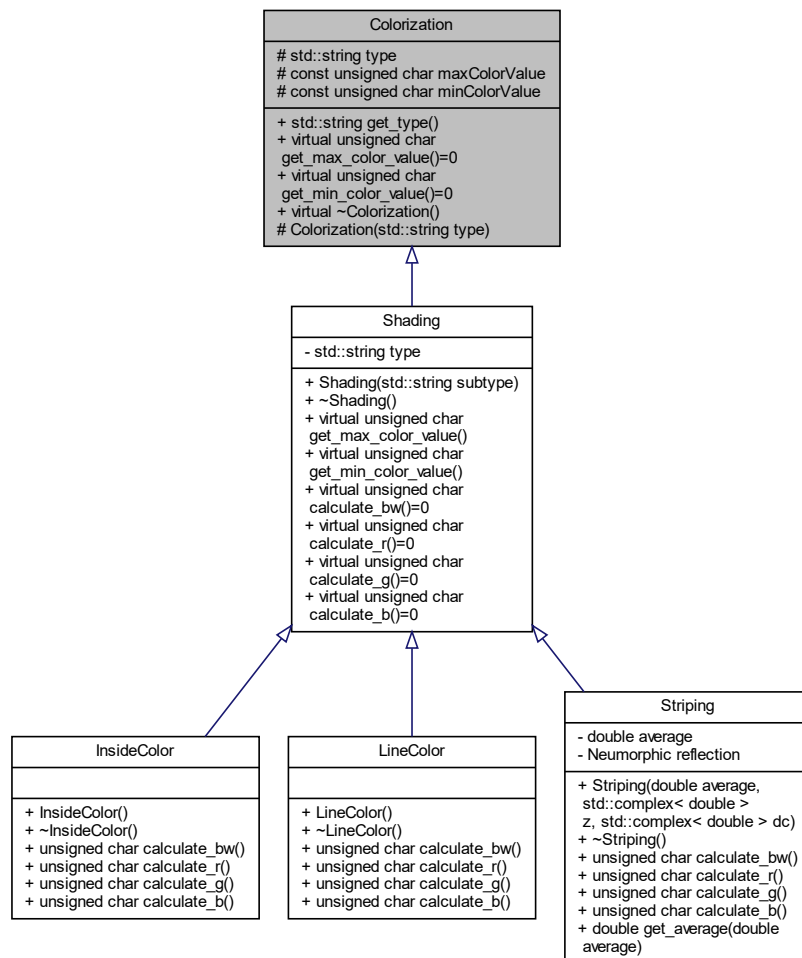
Chapter 5

Class Documentation

5.1 Colorization Class Reference

```
#include <Colorization.h>
```

Inheritance diagram for Colorization:



Collaboration diagram for Colorization:

Colorization
<pre># std::string type # const unsigned char maxColorValue # const unsigned char minColorValue</pre>
<pre>+ std::string get_type() + virtual unsigned char get_max_color_value()=0 + virtual unsigned char get_min_color_value()=0 + virtual ~Colorization() # Colorization(std::string type)</pre>

Public Member Functions

- `std::string` [get_type](#) ()
- virtual unsigned char [get_max_color_value](#) ()=0
- virtual unsigned char [get_min_color_value](#) ()=0
- virtual [~Colorization](#) ()

Protected Member Functions

- [Colorization](#) (std::string [type](#))

Protected Attributes

- `std::string` [type](#)
- const unsigned char [maxColorValue](#) = 255
- const unsigned char [minColorValue](#) = 0

5.1.1 Detailed Description

Definition at line 6 of file [Colorization.h](#).

5.1.2 Constructor & Destructor Documentation

5.1.2.1 ~Colorization()

```
Colorization::~~Colorization ( ) [virtual], [default]
```

5.1.2.2 Colorization()

```
Colorization::Colorization (
    std::string type ) [explicit], [protected]
```

Parametrized constructor

Parameters

<i>type</i>	of colorization
-------------	-----------------

Definition at line 7 of file [Colorization.cpp](#).

```
00007 : type(std::move(type))
00008 {}
```

5.1.3 Member Function Documentation

5.1.3.1 get_max_color_value()

```
virtual unsigned char Colorization::get_max_color_value ( ) [pure virtual]
```

Implemented in [Shading](#).

5.1.3.2 get_min_color_value()

```
virtual unsigned char Colorization::get_min_color_value ( ) [pure virtual]
```

Implemented in [Shading](#).

5.1.3.3 get_type()

```
std::string Colorization::get_type ( )
```

Returns

type of colorization

Definition at line 10 of file [Colorization.cpp](#).

```
00011 {
00012     return type;
00013 }
```

5.1.4 Member Data Documentation

5.1.4.1 maxColorValue

```
const unsigned char Colorization::maxColorValue = 255 [protected]
```

Definition at line 34 of file [Colorization.h](#).

5.1.4.2 minColorValue

```
const unsigned char Colorization::minColorValue = 0 [protected]
```

Definition at line 36 of file [Colorization.h](#).

5.1.4.3 type

```
std::string Colorization::type [protected]
```

Determines type of colorization

Definition at line 32 of file [Colorization.h](#).

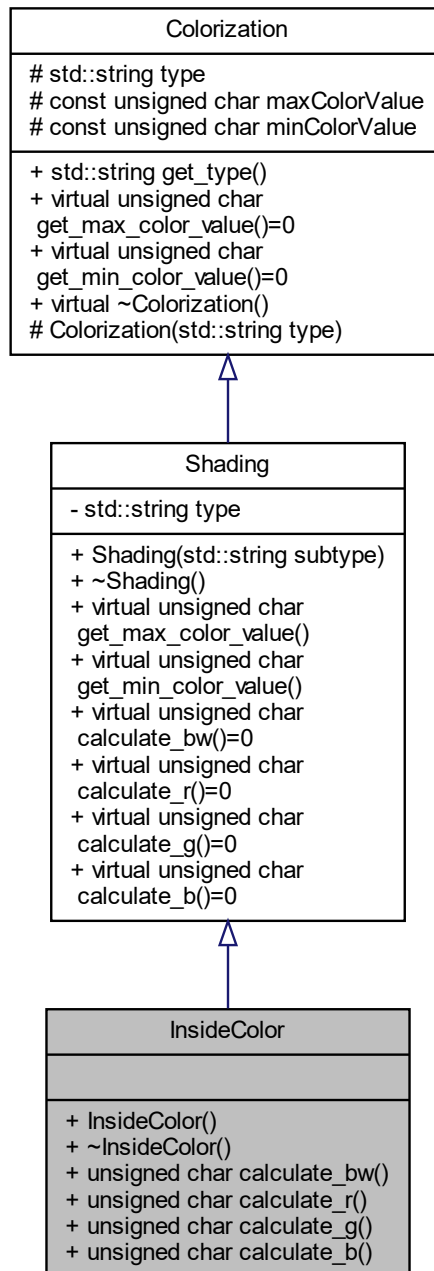
The documentation for this class was generated from the following files:

- [D:/Documents/Final/Mandelbrot_Cpp_Final/src/Colorization.h](#)
- [D:/Documents/Final/Mandelbrot_Cpp_Final/src/Colorization.cpp](#)

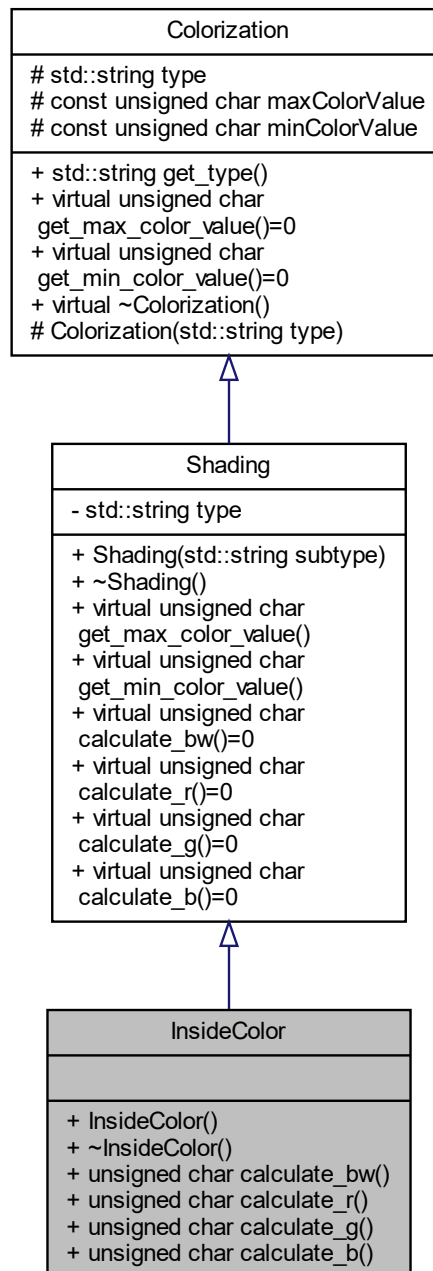
5.2 InsideColor Class Reference

```
#include <InsideColor.h>
```

Inheritance diagram for InsideColor:



Collaboration diagram for InsideColor:



Public Member Functions

- [InsideColor](#) ()
- [~InsideColor](#) ()
- unsigned char [calculate_bw](#) ()
- unsigned char [calculate_r](#) ()
- unsigned char [calculate_g](#) ()
- unsigned char [calculate_b](#) ()

Additional Inherited Members

5.2.1 Detailed Description

Definition at line 6 of file [InsideColor.h](#).

5.2.2 Constructor & Destructor Documentation

5.2.2.1 InsideColor()

```
InsideColor::InsideColor ( )
```

Definition at line 3 of file [InsideColor.cpp](#).

```
00003         : Shading("Inside")
00004 {}
```

5.2.2.2 ~InsideColor()

```
InsideColor::~~InsideColor ( ) [default]
```

5.2.3 Member Function Documentation

5.2.3.1 calculate_b()

```
unsigned char InsideColor::calculate_b ( ) [virtual]
```

Implements [Shading](#).

Definition at line 24 of file [InsideColor.cpp](#).

```
00025 {
00026     return minColorValue;
00027 }
```

5.2.3.2 calculate_bw()

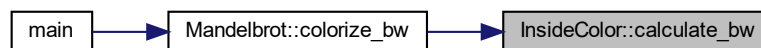
```
unsigned char InsideColor::calculate_bw ( ) [virtual]
```

Implements [Shading](#).

Definition at line 9 of file [InsideColor.cpp](#).

```
00010 {  
00011     return minColorValue;  
00012 }
```

Here is the caller graph for this function:



5.2.3.3 calculate_g()

```
unsigned char InsideColor::calculate_g ( ) [virtual]
```

Implements [Shading](#).

Definition at line 19 of file [InsideColor.cpp](#).

```
00020 {  
00021     return minColorValue;  
00022 }
```

5.2.3.4 calculate_r()

```
unsigned char InsideColor::calculate_r ( ) [virtual]
```

Implements [Shading](#).

Definition at line 14 of file [InsideColor.cpp](#).

```
00015 {  
00016     return minColorValue;  
00017 }
```

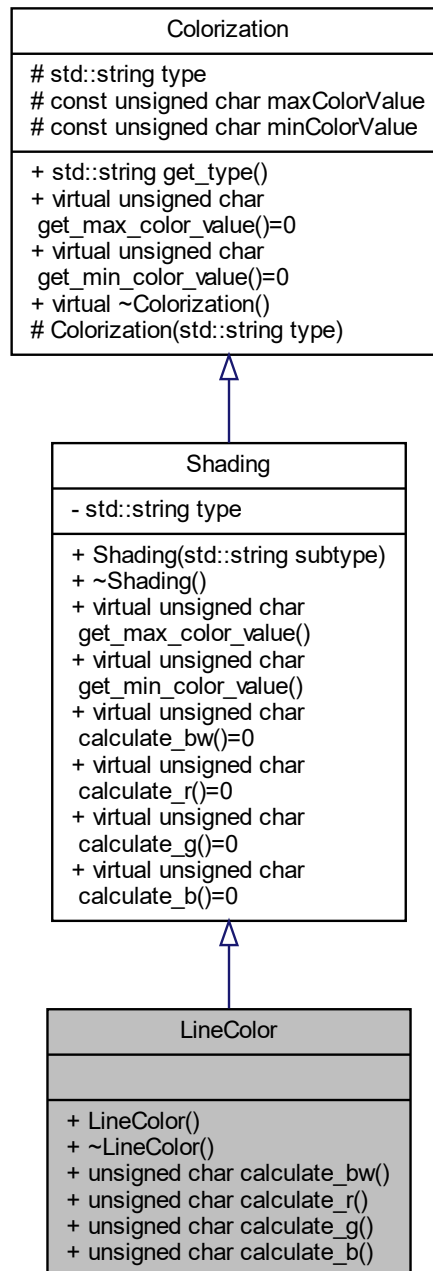
The documentation for this class was generated from the following files:

- D:/Documents/Final/Mandelbrot_Cpp_Final/src/[InsideColor.h](#)
- D:/Documents/Final/Mandelbrot_Cpp_Final/src/[InsideColor.cpp](#)

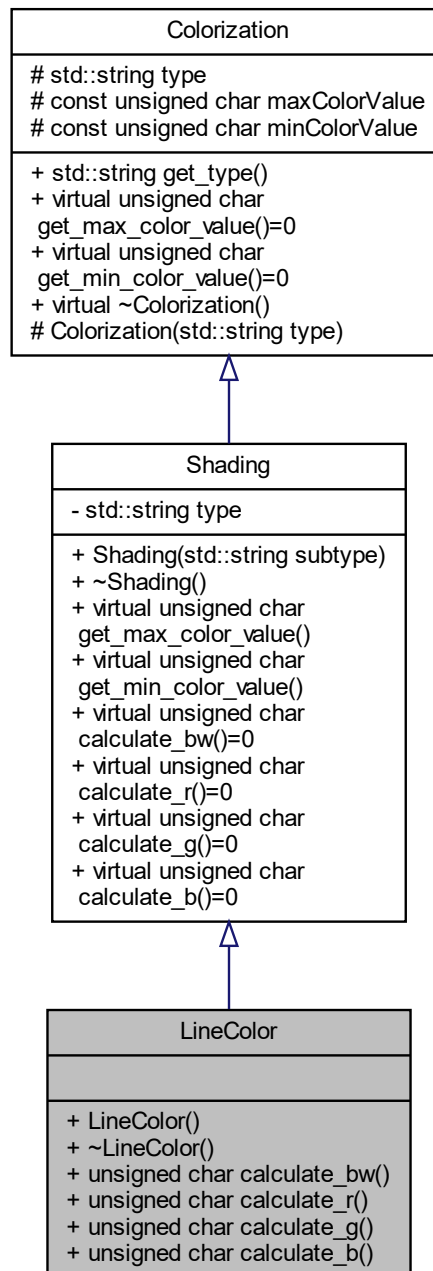
5.3 LineColor Class Reference

```
#include <LineColor.h>
```

Inheritance diagram for LineColor:



Collaboration diagram for LineColor:



Public Member Functions

- [LineColor](#) ()
- [~LineColor](#) ()
- unsigned char [calculate_bw](#) ()
- unsigned char [calculate_r](#) ()
- unsigned char [calculate_g](#) ()
- unsigned char [calculate_b](#) ()

Additional Inherited Members

5.3.1 Detailed Description

Definition at line 6 of file [LineColor.h](#).

5.3.2 Constructor & Destructor Documentation

5.3.2.1 LineColor()

```
LineColor::LineColor ( )
```

Definition at line 3 of file [LineColor.cpp](#).

```
00003         : Shading("Line")
00004 {}
```

5.3.2.2 ~LineColor()

```
LineColor::~LineColor ( ) [default]
```

5.3.3 Member Function Documentation

5.3.3.1 calculate_b()

```
unsigned char LineColor::calculate_b ( ) [virtual]
```

Implements [Shading](#).

Definition at line 24 of file [LineColor.cpp](#).

```
00025 {
00026     return maxColorValue;
00027 }
```

5.3.3.2 calculate_bw()

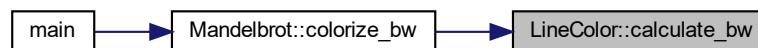
```
unsigned char LineColor::calculate_bw ( ) [virtual]
```

Implements [Shading](#).

Definition at line 9 of file [LineColor.cpp](#).

```
00010 {  
00011     return maxColorValue;  
00012 }
```

Here is the caller graph for this function:



5.3.3.3 calculate_g()

```
unsigned char LineColor::calculate_g ( ) [virtual]
```

Implements [Shading](#).

Definition at line 19 of file [LineColor.cpp](#).

```
00020 {  
00021     return maxColorValue;  
00022 }
```

5.3.3.4 calculate_r()

```
unsigned char LineColor::calculate_r ( ) [virtual]
```

Implements [Shading](#).

Definition at line 14 of file [LineColor.cpp](#).

```
00015 {  
00016     return maxColorValue;  
00017 }
```

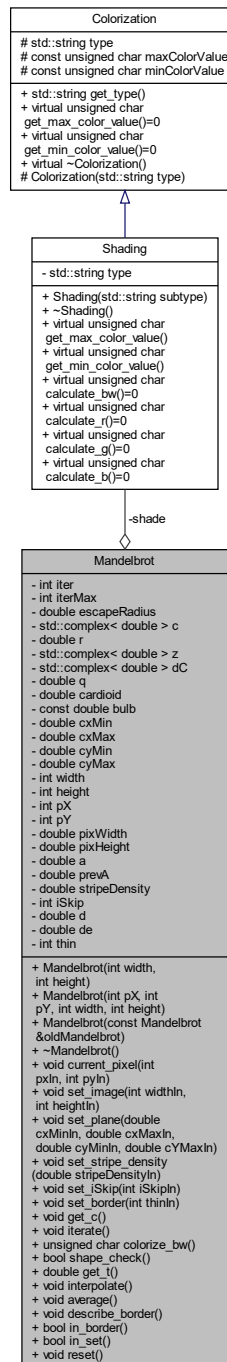
The documentation for this class was generated from the following files:

- D:/Documents/Final/Mandelbrot_Cpp_Final/src/[LineColor.h](#)
- D:/Documents/Final/Mandelbrot_Cpp_Final/src/[LineColor.cpp](#)

5.4 Mandelbrot Class Reference

```
#include <Mandelbrot.h>
```

Collaboration diagram for Mandelbrot:



Public Member Functions

- [Mandelbrot](#) (int [width](#), int [height](#))

- [Mandelbrot](#) (int [pX](#), int [pY](#), int [width](#), int [height](#))
- [Mandelbrot](#) (const [Mandelbrot](#) &oldMandelbrot)
- [~Mandelbrot](#) ()
- void [current_pixel](#) (int [pxIn](#), int [pyIn](#))
- void [set_image](#) (int [widthIn](#), int [heightIn](#))
- void [set_plane](#) (double [cxMinIn](#), double [cxMaxIn](#), double [cyMinIn](#), double [cYMaxIn](#))
- void [set_stripe_density](#) (double [stripeDensityIn](#))
- void [set_iSkip](#) (int [iSkipIn](#))
- void [set_border](#) (int [thinIn](#))
- void [get_c](#) ()
- void [iterate](#) ()
- unsigned char [colorize_bw](#) ()
- bool [shape_check](#) ()
- double [get_t](#) ()
- void [interpolate](#) ()
- void [average](#) ()
- void [describe_border](#) ()
- bool [in_border](#) ()
- bool [in_set](#) ()
- void [reset](#) ()

Private Attributes

- int [iter](#)
- int [iterMax](#)
- double [escapeRadius](#)
- std::complex< double > [c](#)
- double [r](#)
- std::complex< double > [z](#)
- std::complex< double > [dC](#)
- double [q](#)
- double [cardioid](#)
- const double [bulb](#) = 0.0625
- double [cxMin](#)
- double [cxMax](#)
- double [cyMin](#)
- double [cyMax](#)
- int [width](#)
- int [height](#)
- int [pX](#)
- int [pY](#)
- double [pixWidth](#)
- double [pixHeight](#)
- double [a](#)
- double [prevA](#)
- double [stripeDensity](#)
- int [iSkip](#)
- double [d](#)
- double [de](#)
- int [thin](#)
- [Shading](#) * [shade](#)

Friends

- `std::ostream & operator<< (std::ostream &os, const Mandelbrot &mandelbrot)`

5.4.1 Detailed Description

Definition at line 16 of file [Mandelbrot.h](#).

5.4.2 Constructor & Destructor Documentation

5.4.2.1 Mandelbrot() [1/3]

```
Mandelbrot::Mandelbrot (
    int width,
    int height )
```

Default parametrized constructor

Parameters

<i>width</i>	- image width
<i>height</i>	- image height

Definition at line 5 of file [Mandelbrot.cpp](#).

```
00005                                     : width(width), height(height)
00006 {
00007     pX = 0;
00008     pY = 0;
00009     iter = 0;
00010     iterMax = 1000;
00011     escapeRadius = 1000000.0;
00012     // lnER = log(escapeRadius);
00013     c = 0.0;
00014     r = 0.0;
00015     z = 0.0;
00016     dC = 0.0;
00017     q = 0.0;
00018     cardioid = 0.0;
00019     a = 0.0;
00020     prevA = 0.0;
00021     stripeDensity = 7.0;
00022     d = 0.0;
00023     de = 0.0;
00024     cxMin = -2.2;
00025     cxMax = 0.8;
00026     cyMin = -1.5;
00027     cyMax = 1.5;
00028     pixWidth = 0.0;
00029     pixHeight = 0.0;
00030     iSkip = 1;
00031     thin = 3;
00032     shade = nullptr; // avoid calling "new" more than once per pixel
00033 }
```

5.4.2.2 Mandelbrot() [2/3]

```
Mandelbrot::Mandelbrot (
```

```

    int pX,
    int pY,
    int width,
    int height )

```

Minimal parametrized constructor used for calling within pixel loops

Parameters

<i>pX</i>	- current pixel x
<i>pY</i>	- current pixel y
<i>width</i>	- image width
<i>height</i>	- image height

Definition at line 35 of file [Mandelbrot.cpp](#).

```

00035                                     : pX(pX), pY(pY), width(width),
00036 height(height)
00037 {
00038     iter = 0;
00039     iterMax = 1000;
00040     escapeRadius = 1000000.0;
00041     // lnER = log(escapeRadius);
00042     c = 0.0;
00043     r = 0.0;
00044     z = 0.0;
00045     dC = 0.0;
00046     q = 0.0;
00047     cardioid = 0.0;
00048     a = 0.0;
00049     prevA = 0.0;
00050     stripeDensity = 7.0;
00051     d = 0.0;
00052     de = 0.0;
00053     cxMin = -2.2;
00054     cxMax = 0.8;
00055     cyMin = -1.5;
00056     cyMax = 1.5;
00057     pixWidth = 0.0;
00058     pixHeight = 0.0;
00059     iSkip = 1;
00060     thin = 3;
00061     shade = nullptr; // avoid calling "new" more than once per pixel
00062 }

```

5.4.2.3 Mandelbrot() [3/3]

```

Mandelbrot::Mandelbrot (
    const Mandelbrot & oldMandelbrot )

```

Definition at line 256 of file [Mandelbrot.cpp](#).

```

00256                                     : Mandelbrot(oldMandelbrot.width,
00257                                     oldMandelbrot.height)
00258 {}

```

5.4.2.4 ~Mandelbrot()

```

Mandelbrot::~Mandelbrot ( )

```

Definition at line 64 of file [Mandelbrot.cpp](#).

```

00065 {
00066     delete shade;
00067 }

```

5.4.3 Member Function Documentation

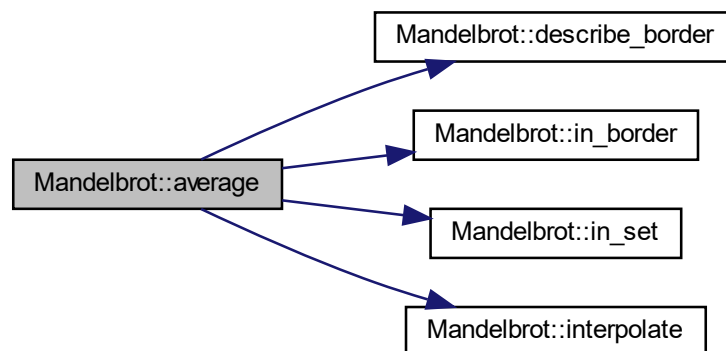
5.4.3.1 average()

```
void Mandelbrot::average ( )
```

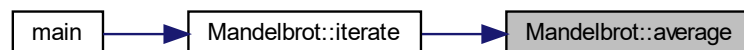
Definition at line 187 of file [Mandelbrot.cpp](#).

```
00188 {  
00189     if (in_set())  
00190     {  
00191         a = -1.0;  
00192     } else {  
00193         describe_border();  
00194         if (in_border()) // in border  
00195         {  
00196             a = FP_ZERO;  
00197         } else {  
00198             a /= static_cast<double>((iter - iSkip)); // A(n)  
00199             prevA /= static_cast<double>((iter - iSkip - 1)); // A(n-1)  
00200             this->interpolate();  
00201             a = (d * a) + ((1.0 - d) * prevA);  
00202         }  
00203     }  
00204 }
```

Here is the call graph for this function:



Here is the caller graph for this function:



5.4.3.2 colorize_bw()

```
unsigned char Mandelbrot::colorize_bw ( )
```

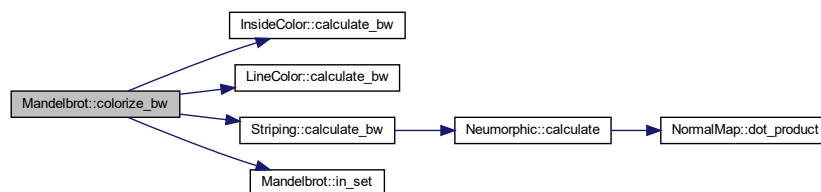
Returns

single output pixel value

Definition at line 140 of file [Mandelbrot.cpp](#).

```
00141 {
00142     if (in_set())
00143     {
00144         shade = new InsideColor();
00145         InsideColor *color;
00146         color = dynamic_cast<InsideColor*>(shade);
00147         return color->calculate_bw();
00148     } else if (a == FP_ZERO) {
00149         shade = new LineColor();
00150         LineColor *color;
00151         color = dynamic_cast<LineColor*>(shade);
00152         return color->calculate_bw();
00153     } else {
00154         shade = new Striping(a, z, dC);
00155         Striping *color;
00156         color = dynamic_cast<Striping*>(shade);
00157         return color->calculate_bw();
00158     }
00159 }
```

Here is the call graph for this function:



Here is the caller graph for this function:



5.4.3.3 current_pixel()

```
void Mandelbrot::current_pixel (
    int pxIn,
    int pyIn )
```

Set current position within image

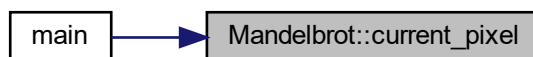
Parameters

$px \leftrightarrow$ <i>In</i>	
$py \leftrightarrow$ <i>In</i>	

Definition at line 75 of file [Mandelbrot.cpp](#).

```
00076 {  
00077     pX = pxIn;  
00078     pY = pyIn;  
00079 }
```

Here is the caller graph for this function:



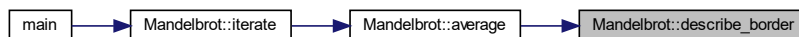
5.4.3.4 describe_border()

```
void Mandelbrot::describe_border ( )
```

Definition at line 206 of file [Mandelbrot.cpp](#).

```
00207 {  
00208     de = 2.0 * r * log(r) / abs(dC);  
00209 }
```

Here is the caller graph for this function:



5.4.3.5 get_c()

```
void Mandelbrot::get_c ( )
```

Determine where pixel lies in complex plane

Definition at line 104 of file [Mandelbrot.cpp](#).

```
00105 {  
00106     pixWidth = (cxMax-cxMin) / static_cast<double>(width);  
00107     pixHeight = (cyMax-cyMin) / static_cast<double>(height);  
00108     c = (cxMin + static_cast<double>(pX) * pixWidth) + ((cyMax - static_cast<double>(pY) * pixHeight) *  
        li);  
00109 }
```

Here is the caller graph for this function:



5.4.3.6 get_t()

```
double Mandelbrot::get_t ( )
```

addend function

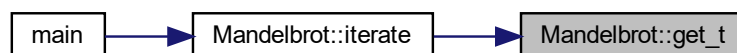
Returns

mapped real number t

Definition at line 174 of file [Mandelbrot.cpp](#).

```
00175 {  
00176     return 0.5 + 0.5 * sin(stripeDensity * arg(z));  
00177 }
```

Here is the caller graph for this function:



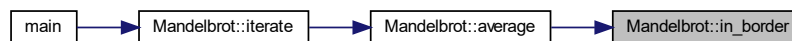
5.4.3.7 in_border()

```
bool Mandelbrot::in_border ( )
```

Definition at line 211 of file [Mandelbrot.cpp](#).

```
00212 {  
00213     if (de < (pixWidth / static_cast<double>(thin)))  
00214     {  
00215         return true;  
00216     } else {  
00217         return false;  
00218     }  
00219 }
```

Here is the caller graph for this function:



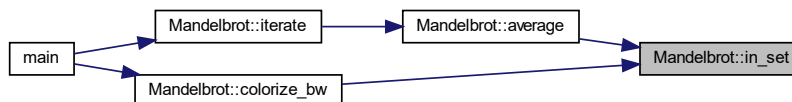
5.4.3.8 in_set()

```
bool Mandelbrot::in_set ( )
```

Definition at line 221 of file [Mandelbrot.cpp](#).

```
00222 {  
00223     if (iter == iterMax)  
00224     {  
00225         return true;  
00226     } else {  
00227         return false;  
00228     }  
00229 }
```

Here is the caller graph for this function:



5.4.3.9 interpolate()

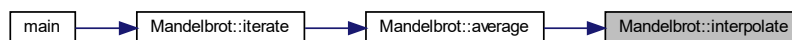
```
void Mandelbrot::interpolate ( )
```

Removes level sets of escape time

Definition at line 179 of file [Mandelbrot.cpp](#).

```
00180 {
00181     // smooth iteration count
00182     d = static_cast<double>(iter + 1) + log(log(escapeRadius) / log(r)) / M_LN2;
00183     d = d - static_cast<double>(static_cast<int>(d)); // only fractional part = interpolation
00184     // coefficient
00185 }
```

Here is the caller graph for this function:



5.4.3.10 iterate()

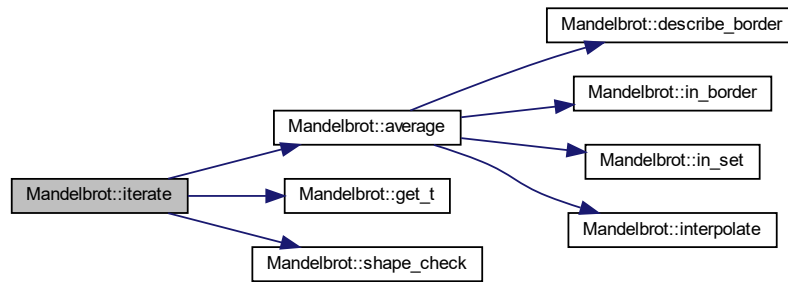
```
void Mandelbrot::iterate ( )
```

Main [Mandelbrot](#) function

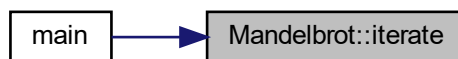
Definition at line 111 of file [Mandelbrot.cpp](#).

```
00112 {
00113     if (!this->shape_check())
00114     {
00115         for (iter = 0; iter < iterMax; iter++)
00116         {
00117             // mandelbrot set formula
00118             dC = 2.0 * dC * z + 1.0;
00119             z = z * z + C;
00120
00121             // compute average
00122             if (iter > iSkip)
00123             {
00124                 a += get_t();
00125             }
00126
00127             r = abs(z);
00128             if (r > escapeRadius)
00129             {
00130                 break;
00131             }
00132
00133             prevA = a;
00134         }
00135
00136         average();
00137     }
00138 }
```

Here is the call graph for this function:



Here is the caller graph for this function:



5.4.3.11 reset()

```
void Mandelbrot::reset ( )
```

Definition at line 242 of file [Mandelbrot.cpp](#).

```

00243 {
00244     c = 0.0;
00245     r = 0.0;
00246     z = 0.0;
00247     dC = 0.0;
00248     q = 0.0;
00249     cardioid = 0.0;
00250     a = 0.0;
00251     prevA = 0.0;
00252     d = 0.0;
00253     shade = nullptr; // avoid calling "new" more than once per pixel
00254 }
```

Here is the caller graph for this function:



5.4.3.12 set_border()

```
void Mandelbrot::set_border (
    int thinIn )
```

Definition at line 99 of file [Mandelbrot.cpp](#).

```
00100 {
00101     thin = thinIn;
00102 }
```

5.4.3.13 set_image()

```
void Mandelbrot::set_image (
    int widthIn,
    int heightIn )
```

Definition at line 69 of file [Mandelbrot.cpp](#).

```
00070 {
00071     width = widthIn;
00072     height = heightIn;
00073 }
```

5.4.3.14 set_iSkip()

```
void Mandelbrot::set_iSkip (
    int iSkipIn )
```

Definition at line 94 of file [Mandelbrot.cpp](#).

```
00095 {
00096     iSkip = iSkipIn;
00097 }
```

5.4.3.15 set_plane()

```
void Mandelbrot::set_plane (
    double cxMinIn,
    double cxMaxIn,
    double cyMinIn,
    double cYMaxIn )
```

Definition at line 81 of file [Mandelbrot.cpp](#).

```
00082 {
00083     cxMin = cxMinIn;
00084     cxMax = cxMaxIn;
00085     cyMin = cyMinIn;
00086     cyMax = cYMaxIn;
00087 }
```

5.4.3.16 set_stripe_density()

```
void Mandelbrot::set_stripe_density (
    double stripeDensityIn )
```

Definition at line 89 of file [Mandelbrot.cpp](#).

```
00090 {
00091     stripeDensity = stripeDensityIn;
00092 }
```

5.4.3.17 shape_check()

```
bool Mandelbrot::shape_check ( )
```

Shape checking algorithm - determines if point is within main cardioid or secondary bulb. Removes about 91% of the set from being iterated. Should not be implemented for a render that does not include these parts, will add unnecessary computing

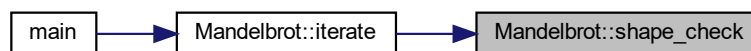
Returns

TRUE if within the main shapes

Definition at line 161 of file [Mandelbrot.cpp](#).

```
00162 {
00163     q = ((real(c) - 0.25) * (real(c) - 0.25)) + (imag(c) * imag(c));
00164     cardioid = 0.25 * imag(c) * imag(c);
00165     if ((real(c) * real(c) + 2.0 * real(c) + 1.0 + imag(c) * imag(c)) < bulb ||
00166         (q * (q + (real(c) - 0.25)) < cardioid))
00167     {
00168         return true;
00169     } else {
00170         return false;
00171     }
00172 }
```

Here is the caller graph for this function:



5.4.4 Friends And Related Function Documentation

5.4.4.1 operator<<

```
std::ostream & operator<< (
    std::ostream & os,
    const Mandelbrot & mandelbrot ) [friend]
```

5.4.5 Member Data Documentation

5.4.5.1 a

```
double Mandelbrot::a [private]
```

Definition at line 156 of file [Mandelbrot.h](#).

5.4.5.2 bulb

```
const double Mandelbrot::bulb = 0.0625 [private]
```

independent

Definition at line 131 of file [Mandelbrot.h](#).

5.4.5.3 c

```
std::complex<double> Mandelbrot::c [private]
```

Definition at line 116 of file [Mandelbrot.h](#).

5.4.5.4 cardioid

```
double Mandelbrot::cardioid [private]
```

Definition at line 126 of file [Mandelbrot.h](#).

5.4.5.5 cxMax

```
double Mandelbrot::cxMax [private]
```

Definition at line 136 of file [Mandelbrot.h](#).

5.4.5.6 cxMin

```
double Mandelbrot::cxMin [private]
```

Definition at line 134 of file [Mandelbrot.h](#).

5.4.5.7 cyMax

```
double Mandelbrot::cyMax [private]
```

Definition at line 140 of file [Mandelbrot.h](#).

5.4.5.8 cyMin

```
double Mandelbrot::cyMin [private]
```

Definition at line 138 of file [Mandelbrot.h](#).

5.4.5.9 d

```
double Mandelbrot::d [private]
```

Definition at line 171 of file [Mandelbrot.h](#).

5.4.5.10 dC

```
std::complex<double> Mandelbrot::dC [private]
```

Definition at line 122 of file [Mandelbrot.h](#).

5.4.5.11 de

```
double Mandelbrot::de [private]
```

Definition at line 174 of file [Mandelbrot.h](#).

5.4.5.12 escapeRadius

```
double Mandelbrot::escapeRadius [private]
```

Definition at line 112 of file [Mandelbrot.h](#).

5.4.5.13 height

```
int Mandelbrot::height [private]
```

Definition at line 145 of file [Mandelbrot.h](#).

5.4.5.14 iSkip

```
int Mandelbrot::iSkip [private]
```

Exclude iSkip+1 elements from average

Definition at line 168 of file [Mandelbrot.h](#).

5.4.5.15 iter

```
int Mandelbrot::iter [private]
```

Definition at line 108 of file [Mandelbrot.h](#).

5.4.5.16 iterMax

```
int Mandelbrot::iterMax [private]
```

Definition at line 110 of file [Mandelbrot.h](#).

5.4.5.17 pixHeight

```
double Mandelbrot::pixHeight [private]
```

Definition at line 153 of file [Mandelbrot.h](#).

5.4.5.18 pixWidth

```
double Mandelbrot::pixWidth [private]
```

Definition at line 151 of file [Mandelbrot.h](#).

5.4.5.19 prevA

```
double Mandelbrot::prevA [private]
```

Definition at line 158 of file [Mandelbrot.h](#).

5.4.5.20 pX

```
int Mandelbrot::pX [private]
```

Definition at line 147 of file [Mandelbrot.h](#).

5.4.5.21 pY

```
int Mandelbrot::pY [private]
```

Definition at line 149 of file [Mandelbrot.h](#).

5.4.5.22 q

```
double Mandelbrot::q [private]
```

Definition at line 124 of file [Mandelbrot.h](#).

5.4.5.23 r

```
double Mandelbrot::r [private]
```

Definition at line 118 of file [Mandelbrot.h](#).

5.4.5.24 shade

```
Shading* Mandelbrot::shade [private]
```

Definition at line 177 of file [Mandelbrot.h](#).

5.4.5.25 stripeDensity

```
double Mandelbrot::stripeDensity [private]
```

Higher is more dense

Definition at line 163 of file [Mandelbrot.h](#).

5.4.5.26 thin

```
int Mandelbrot::thin [private]
```

Definition at line 175 of file [Mandelbrot.h](#).

5.4.5.27 width

```
int Mandelbrot::width [private]
```

Definition at line 143 of file [Mandelbrot.h](#).

5.4.5.28 z

```
std::complex<double> Mandelbrot::z [private]
```

Definition at line 120 of file [Mandelbrot.h](#).

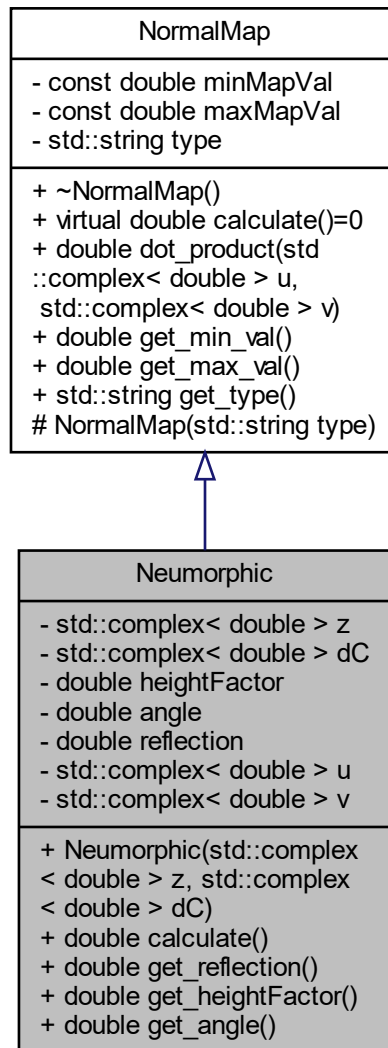
The documentation for this class was generated from the following files:

- D:/Documents/Final/Mandelbrot_Cpp_Final/src/[Mandelbrot.h](#)
- D:/Documents/Final/Mandelbrot_Cpp_Final/src/[Mandelbrot.cpp](#)

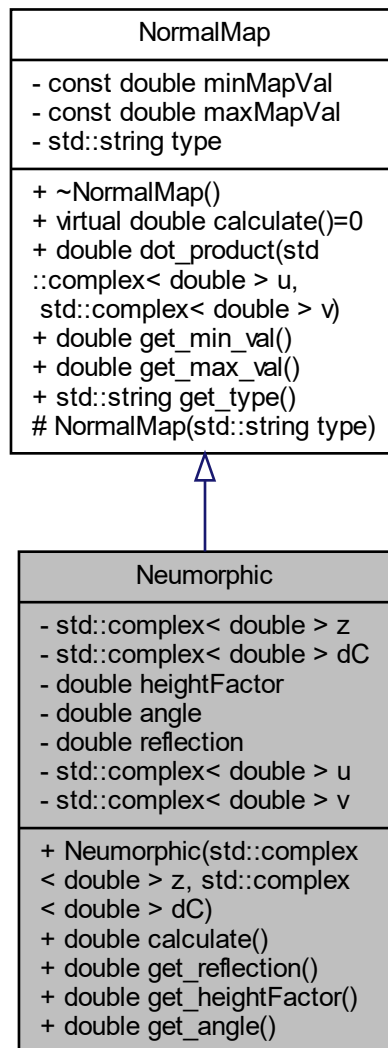
5.5 Neumorphic Class Reference

```
#include <Neumorphic.h>
```

Inheritance diagram for Neumorphic:



Collaboration diagram for Neumorphic:



Public Member Functions

- [Neumorphic](#) (std::complex< double > [z](#), std::complex< double > [dC](#))
- double [calculate](#) ()
- double [get_reflection](#) ()
- double [get_heightFactor](#) ()
- double [get_angle](#) ()

Private Attributes

- std::complex< double > [z](#)
- std::complex< double > [dC](#)

- double [heightFactor](#)
- double [angle](#)
- double [reflection](#)
- `std::complex< double > u`
- `std::complex< double > v`

Additional Inherited Members

5.5.1 Detailed Description

Definition at line 6 of file [Neumorphic.h](#).

5.5.2 Constructor & Destructor Documentation

5.5.2.1 Neumorphic()

```
Neumorphic::Neumorphic (
    std::complex< double > z,
    std::complex< double > dC )
```

Default parametrized constructor

Parameters

<i>z</i>	
<i>dC</i>	

Definition at line 5 of file [Neumorphic.cpp](#).

```
00005                                     : NormalMap("Neumorphic"), z(z), dC
00006 (dC)
00007 {
00008     heightFactor = 1.5;
00009     angle = 45.0 / 360.0;
00010     reflection = FP_ZERO;
00011     v = exp(2.0 * angle * M\_PI * 1i);
00012 }
```

5.5.3 Member Function Documentation

5.5.3.1 calculate()

```
double Neumorphic::calculate ( ) [virtual]
```

Implements [NormalMap](#).

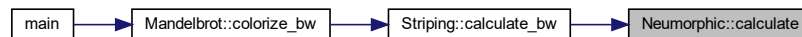
Definition at line 14 of file [Neumorphic.cpp](#).

```
00015 {
00016     u = z / dC;
00017     u = u / abs(u); // normalize
00018     reflection = dot_product(u, v) + heightFactor;
00019     reflection = reflection / (1.0 + heightFactor); // rescale so that it does not get bigger than 1
00020     if (reflection < 0.0)
00021     {
00022         reflection = 0.0;
00023     } else {}
00024     return reflection;
00025 }
```

Here is the call graph for this function:



Here is the caller graph for this function:



5.5.3.2 get_angle()

```
double Neumorphic::get_angle ( )
```

Definition at line 37 of file [Neumorphic.cpp](#).

```
00038 {
00039     return angle;
00040 }
```

5.5.3.3 get_heightFactor()

```
double Neumorphic::get_heightFactor ( )
```

Definition at line 32 of file [Neumorphic.cpp](#).

```
00033 {
00034     return heightFactor;
00035 }
```


5.5.3.4 get_reflection()

```
double Neumorphic::get_reflection ( )
```

Definition at line 27 of file [Neumorphic.cpp](#).

```
00028 {  
00029     return reflection;  
00030 }
```

5.5.4 Member Data Documentation

5.5.4.1 angle

```
double Neumorphic::angle [private]
```

incoming direction of light WRT +x-axis (degrees) change first number

Definition at line 40 of file [Neumorphic.h](#).

5.5.4.2 dC

```
std::complex<double> Neumorphic::dC [private]
```

Definition at line 29 of file [Neumorphic.h](#).

5.5.4.3 heightFactor

```
double Neumorphic::heightFactor [private]
```

of pseudo incoming light vector

Definition at line 34 of file [Neumorphic.h](#).

5.5.4.4 reflection

```
double Neumorphic::reflection [private]
```

normalized normal vector

Definition at line 45 of file [Neumorphic.h](#).

5.5.4.5 u

```
std::complex<double> Neumorphic::u [private]
```

Definition at line 47 of file [Neumorphic.h](#).

5.5.4.6 v

```
std::complex<double> Neumorphic::v [private]
```

unit vector in direction of this.angle

Definition at line 52 of file [Neumorphic.h](#).

5.5.4.7 z

```
std::complex<double> Neumorphic::z [private]
```

Definition at line 27 of file [Neumorphic.h](#).

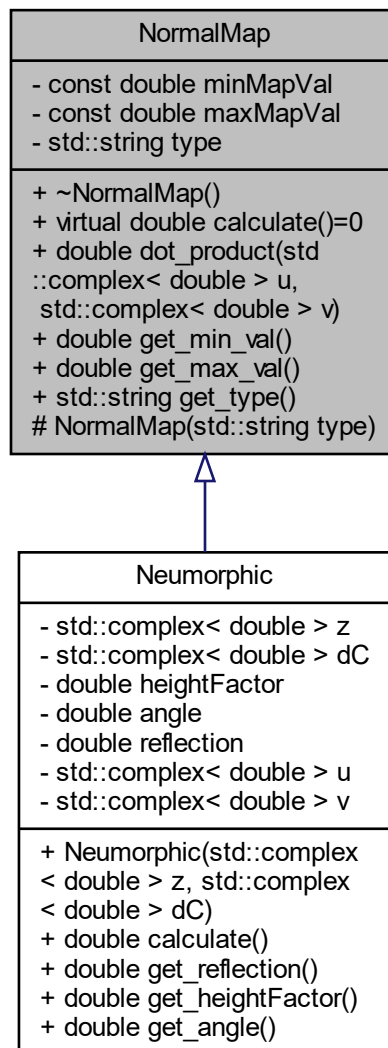
The documentation for this class was generated from the following files:

- D:/Documents/Final/Mandelbrot_Cpp_Final/src/[Neumorphic.h](#)
- D:/Documents/Final/Mandelbrot_Cpp_Final/src/[Neumorphic.cpp](#)

5.6 NormalMap Class Reference

```
#include <NormalMap.h>
```

Inheritance diagram for NormalMap:



Collaboration diagram for NormalMap:

NormalMap
<ul style="list-style-type: none">- const double minMapVal- const double maxMapVal- std::string type
<ul style="list-style-type: none">+ ~NormalMap()+ virtual double calculate()=0+ double dot_product(std::complex< double > u, std::complex< double > v)+ double get_min_val()+ double get_max_val()+ std::string get_type()# NormalMap(std::string type)

Public Member Functions

- [~NormalMap](#) ()
- virtual double [calculate](#) ()=0
- double [dot_product](#) (std::complex< double > u, std::complex< double > v)
- double [get_min_val](#) ()
- double [get_max_val](#) ()
- std::string [get_type](#) ()

Protected Member Functions

- [NormalMap](#) (std::string [type](#))

Private Attributes

- const double [minMapVal](#) = 0.0
- const double [maxMapVal](#) = 1.0
- std::string [type](#)

5.6.1 Detailed Description

Definition at line 8 of file [NormalMap.h](#).

5.6.2 Constructor & Destructor Documentation

5.6.2.1 ~NormalMap()

```
NormalMap::~~NormalMap ( ) [default]
```

5.6.2.2 NormalMap()

```
NormalMap::NormalMap (
    std::string type ) [explicit], [protected]
```

Definition at line 5 of file [NormalMap.cpp](#).

```
00006 {}
```

5.6.3 Member Function Documentation

5.6.3.1 calculate()

```
virtual double NormalMap::calculate ( ) [pure virtual]
```

Implemented in [Neumorphic](#).

5.6.3.2 dot_product()

```
double NormalMap::dot_product (
    std::complex< double > u,
    std::complex< double > v )
```

Complex number dot product

Parameters

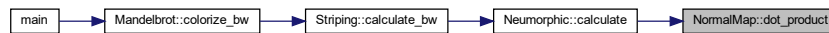
<i>u</i>	
<i>v</i>	

Returns

Definition at line 11 of file [NormalMap.cpp](#).

```
00012 {
00013     return real(u) * real(v) + imag(u) * imag(v);
00014 }
```

Here is the caller graph for this function:



5.6.3.3 get_max_val()

```
double NormalMap::get_max_val ( )
```

Definition at line 21 of file [NormalMap.cpp](#).

```
00022 {  
00023     return maxMapVal;  
00024 }
```

5.6.3.4 get_min_val()

```
double NormalMap::get_min_val ( )
```

Definition at line 16 of file [NormalMap.cpp](#).

```
00017 {  
00018     return minMapVal;  
00019 }
```

5.6.3.5 get_type()

```
std::string NormalMap::get_type ( )
```

Definition at line 25 of file [NormalMap.cpp](#).

```
00026 {  
00027     return type;  
00028 }
```

5.6.4 Member Data Documentation

5.6.4.1 maxMapVal

```
const double NormalMap::maxMapVal = 1.0 [private]
```

Definition at line 38 of file [NormalMap.h](#).

5.6.4.2 minMapVal

```
const double NormalMap::minMapVal = 0.0 [private]
```

Definition at line 36 of file [NormalMap.h](#).

5.6.4.3 type

```
std::string NormalMap::type [private]
```

Definition at line 40 of file [NormalMap.h](#).

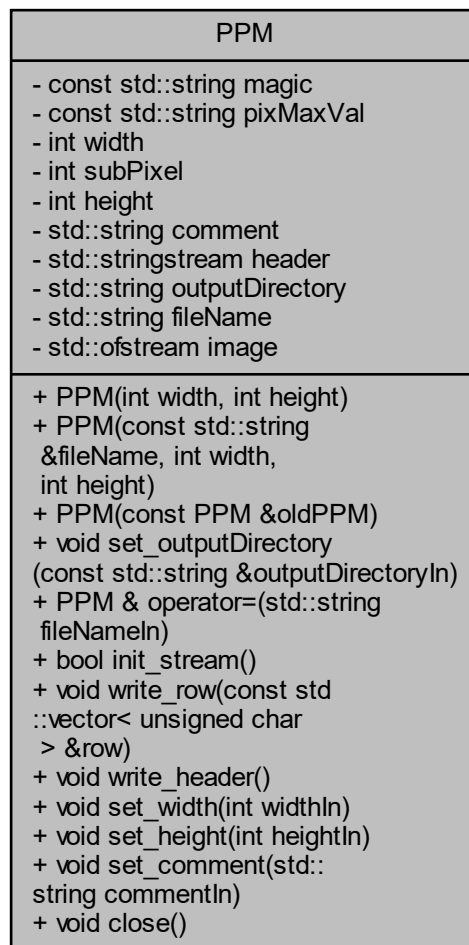
The documentation for this class was generated from the following files:

- D:/Documents/Final/Mandelbrot_Cpp_Final/src/[NormalMap.h](#)
- D:/Documents/Final/Mandelbrot_Cpp_Final/src/[NormalMap.cpp](#)

5.7 PPM Class Reference

```
#include <PPM.h>
```

Collaboration diagram for PPM:



Public Member Functions

- [PPM](#) (int [width](#), int [height](#))
- [PPM](#) (const std::string &[fileName](#), int [width](#), int [height](#))
- [PPM](#) (const [PPM](#) &oldPPM)
- void [set_outputDirectory](#) (const std::string &outputDirectoryIn)
- [PPM](#) & [operator=](#) (std::string fileNameIn)
- bool [init_stream](#) ()
- void [write_row](#) (const std::vector< unsigned char > &row)
- void [write_header](#) ()
- void [set_width](#) (int widthIn)
- void [set_height](#) (int heightIn)
- void [set_comment](#) (std::string commentIn)
- void [close](#) ()

Private Attributes

- const std::string `magic` = "P6\n"
- const std::string `pixMaxVal` = "255\n"
- int `width`
- int `subPixel`
- int `height`
- std::string `comment`
- std::stringstream `header`
- std::string `outputDirectory`
- std::string `fileName`
- std::ofstream `image`

5.7.1 Detailed Description

Definition at line 11 of file `PPM.h`.

5.7.2 Constructor & Destructor Documentation

5.7.2.1 PPM() [1/3]

```
PPM::PPM (
    int width,
    int height )
```

Default parametrized constructor contains output directory and file name

Parameters

<i>width</i>	
<i>height</i>	

Definition at line 7 of file `PPM.cpp`.

```
00007                                     : width(width), height(height)
00008 {
00009     outputDirectory = "..\\output\\";
00010     fileName = outputDirectory + "gigabrot_default.ppm";
00011     subPixel = width * 3;
00012 }
```

5.7.2.2 PPM() [2/3]

```
PPM::PPM (
    const std::string & fileName,
    int width,
    int height )
```

Minimal parametrized constructor Only needs file name, contains output directory

Parameters

<i>fileName</i>	
<i>width</i>	
<i>height</i>	

Definition at line 14 of file [PPM.cpp](#).

```

00014                                     : width(width), height(height)
00015 {
00016     outputDirectory = "..\\output\\";
00017     this->fileName = outputDirectory + fileName;
00018     subPixel = width * 3;
00019 }
```

5.7.2.3 PPM() [3/3]

```

PPM::PPM (
    const PPM & oldPPM )
```

Copy constructor

Parameters

<i>oldPPM</i>	- dereferenced old object
---------------	---------------------------

Definition at line 21 of file [PPM.cpp](#).

```

00021                                     : magic(oldPPM.magic), pixMaxVal(oldPPM.pixMaxVal),
00022     width(oldPPM.width), height(oldPPM.height), subPixel(oldPPM.subPixel), comment(oldPPM.comment)
00023 {}
```

5.7.3 Member Function Documentation

5.7.3.1 close()

```
void PPM::close ( )
```

Close ofstream for image

Definition at line 72 of file [PPM.cpp](#).

```

00073 {
00074     image.close();
00075     cout << "File " << fileName << " saved\n";
00076 }
```

Here is the caller graph for this function:



5.7.3.2 init_stream()

```
bool PPM::init_stream ( )
```

Initialize ofstream

Returns

Definition at line 36 of file [PPM.cpp](#).

```
00037 {
00038     image.open(fileName, ios::binary);
00039
00040     if (image.is_open())
00041     {
00042         return true;
00043     } else
00044     {
00045         return false;
00046     }
00047 }
```

Here is the caller graph for this function:



5.7.3.3 operator=()

```
PPM & PPM::operator= (
    std::string fileNameIn )
```

Overloaded assignment operator used to override output directory and file name

Parameters

<i>file</i> ↔	
<i>fileNameIn</i>	

Returns

Definition at line 30 of file [PPM.cpp](#).

```
00031 {
00032     this->fileName = std::move(fileNameIn);
00033     return *this;
00034 }
```

5.7.3.4 set_comment()

```
void PPM::set_comment (
    std::string commentIn )
```

Set custom comment for [PPM](#) image

Parameters

<i>commentIn</i>	
------------------	--

Definition at line 67 of file [PPM.cpp](#).

```
00068 {
00069     comment = std::move(commentIn);
00070 }
```

5.7.3.5 set_height()

```
void PPM::set_height (
    int heightIn )
```

Set height of [PPM](#) image

Parameters

<i>heightIn</i>	
-----------------	--

Definition at line 62 of file [PPM.cpp](#).

```
00063 {
00064     height = heightIn;
00065 }
```

5.7.3.6 set_outputDirectory()

```
void PPM::set_outputDirectory (
    const std::string & outputDirectoryIn )
```

Sets output directory destination

Parameters

<i>file</i> ↔	
<i>NameIn</i>	

Definition at line 25 of file [PPM.cpp](#).

```
00026 {
00027     outputDirectory = outputDirectoryIn;
00028 }
```

5.7.3.7 set_width()

```
void PPM::set_width (
    int widthIn )
```

Set width of [PPM](#) image

Parameters

<i>width</i> ↔	
<i>In</i>	

Definition at line 57 of file [PPM.cpp](#).

```
00058 {
00059     width = widthIn;
00060 }
```

5.7.3.8 write_header()

```
void PPM::write_header ( )
```

Print header data to [PPM](#) file

Definition at line 49 of file [PPM.cpp](#).

```
00050 {
00051     string widthStr = to_string(this->width);
00052     string lengthStr = to_string(this->height);
00053     header << magic << widthStr << " " << lengthStr << "\n" << comment << "\n" << pixMaxVal;
00054     image << header.rdbuf();
00055 }
```

Here is the caller graph for this function:



5.7.3.9 write_row()

```
void PPM::write_row (
    const std::vector< unsigned char > & row ) [inline]
```

Print row of pixels to [PPM](#) file - templated for size of array

Template Parameters

<i>N</i>	- std::array size
----------	-------------------

Parameters

<i>row</i>	- std::array of pixels
------------	------------------------

Definition at line [63](#) of file [PPM.h](#).

```
00064 {
00065     image.write((char const *) row.data(), row.size());
00066 }
```

Here is the caller graph for this function:



5.7.4 Member Data Documentation

5.7.4.1 comment

```
std::string PPM::comment [private]
```

Definition at line [112](#) of file [PPM.h](#).

5.7.4.2 fileName

```
std::string PPM::fileName [private]
```

Definition at line [121](#) of file [PPM.h](#).

5.7.4.3 header

```
std::stringstream PPM::header [private]
```

Contains full header info

Definition at line 117 of file [PPM.h](#).

5.7.4.4 height

```
int PPM::height [private]
```

Definition at line 110 of file [PPM.h](#).

5.7.4.5 image

```
std::ofstream PPM::image [private]
```

Definition at line 123 of file [PPM.h](#).

5.7.4.6 magic

```
const std::string PPM::magic = "P6\n" [private]
```

Determines type of [PPM](#)

Definition at line 102 of file [PPM.h](#).

5.7.4.7 outputDirectory

```
std::string PPM::outputDirectory [private]
```

Definition at line 119 of file [PPM.h](#).

5.7.4.8 pixMaxVal

```
const std::string PPM::pixMaxVal = "255\n" [private]
```

Definition at line 104 of file [PPM.h](#).

5.7.4.9 subPixel

```
int PPM::subPixel [private]
```

Definition at line 108 of file [PPM.h](#).

5.7.4.10 width

```
int PPM::width [private]
```

Definition at line 106 of file [PPM.h](#).

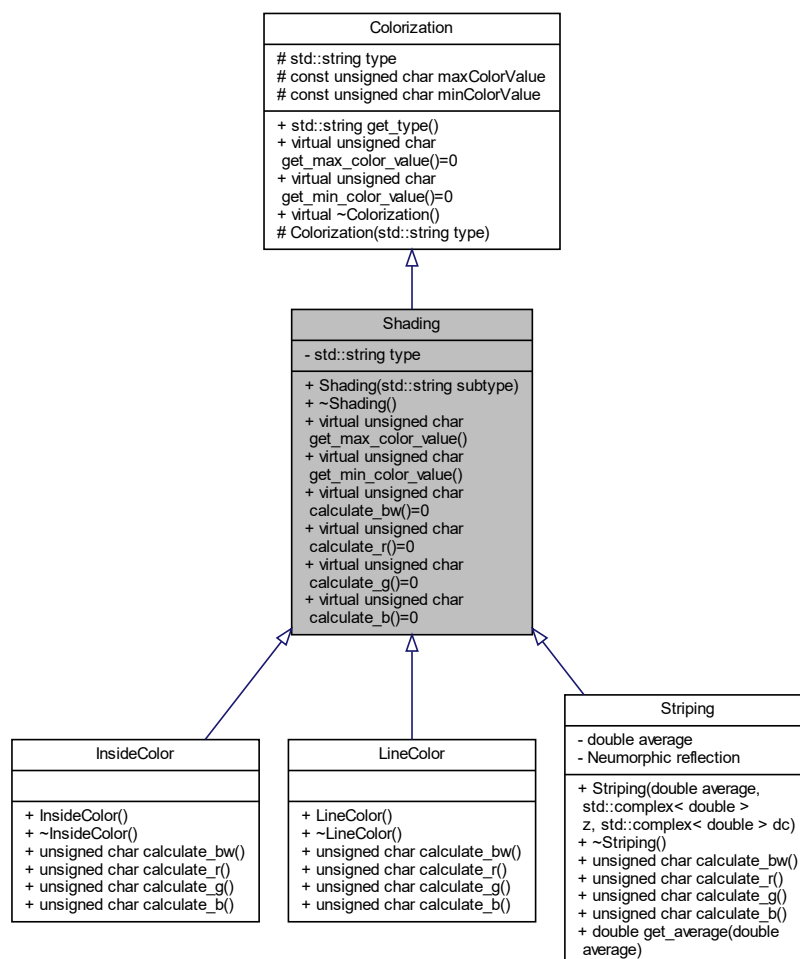
The documentation for this class was generated from the following files:

- [D:/Documents/Final/Mandelbrot_Cpp_Final/src/PPM.h](#)
- [D:/Documents/Final/Mandelbrot_Cpp_Final/src/PPM.cpp](#)

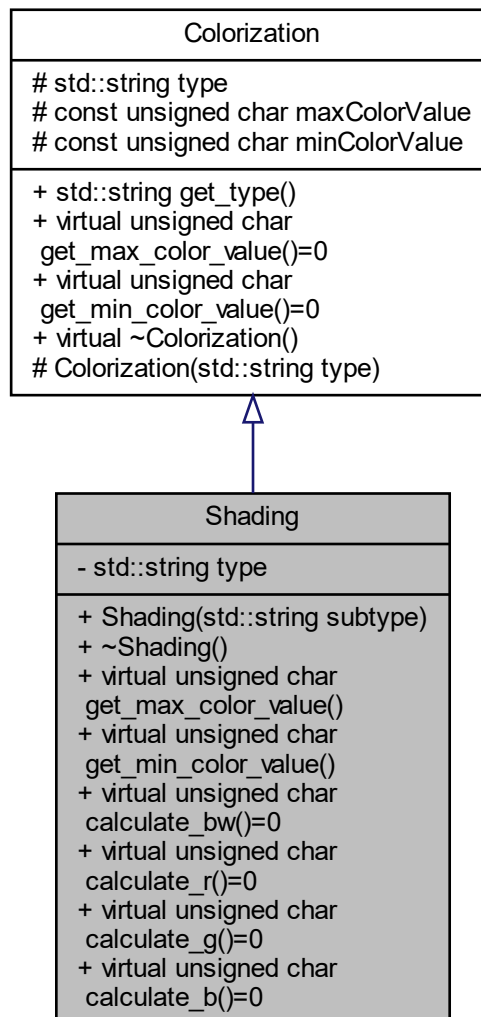
5.8 Shading Class Reference

```
#include <Shading.h>
```

Inheritance diagram for Shading:



Collaboration diagram for Shading:



Public Member Functions

- [Shading](#) (std::string subtype)
- [~Shading](#) ()
- virtual unsigned char [get_max_color_value](#) ()
- virtual unsigned char [get_min_color_value](#) ()
- virtual unsigned char [calculate_bw](#) ()=0
- virtual unsigned char [calculate_r](#) ()=0
- virtual unsigned char [calculate_g](#) ()=0
- virtual unsigned char [calculate_b](#) ()=0

Private Attributes

- std::string [type](#) = "Shading"

Additional Inherited Members

5.8.1 Detailed Description

Definition at line 6 of file [Shading.h](#).

5.8.2 Constructor & Destructor Documentation

5.8.2.1 Shading()

```
Shading::Shading (
    std::string subtype ) [explicit]
```

Definition at line 5 of file [Shading.cpp](#).

```
00005         : Colorization(std::move(type))
00006 {}
```

5.8.2.2 ~Shading()

```
Shading::~Shading ( ) [default]
```

5.8.3 Member Function Documentation

5.8.3.1 calculate_b()

```
virtual unsigned char Shading::calculate_b ( ) [pure virtual]
```

Implemented in [InsideColor](#), [LineColor](#), and [Striping](#).

5.8.3.2 calculate_bw()

```
virtual unsigned char Shading::calculate_bw ( ) [pure virtual]
```

Implemented in [InsideColor](#), [LineColor](#), and [Striping](#).

5.8.3.3 calculate_g()

```
virtual unsigned char Shading::calculate_g ( ) [pure virtual]
```

Implemented in [InsideColor](#), [LineColor](#), and [Striping](#).

5.8.3.4 calculate_r()

```
virtual unsigned char Shading::calculate_r ( ) [pure virtual]
```

Implemented in [InsideColor](#), [LineColor](#), and [Striping](#).

5.8.3.5 get_max_color_value()

```
unsigned char Shading::get_max_color_value ( ) [virtual]
```

Implements [Colorization](#).

Definition at line 11 of file [Shading.cpp](#).

```
00012 {  
00013     return maxColorValue;  
00014 }
```

5.8.3.6 get_min_color_value()

```
unsigned char Shading::get_min_color_value ( ) [virtual]
```

Implements [Colorization](#).

Definition at line 16 of file [Shading.cpp](#).

```
00017 {  
00018     return minColorValue;  
00019 }
```

5.8.4 Member Data Documentation

5.8.4.1 type

```
std::string Shading::type = "Shading" [private]
```

Definition at line 29 of file [Shading.h](#).

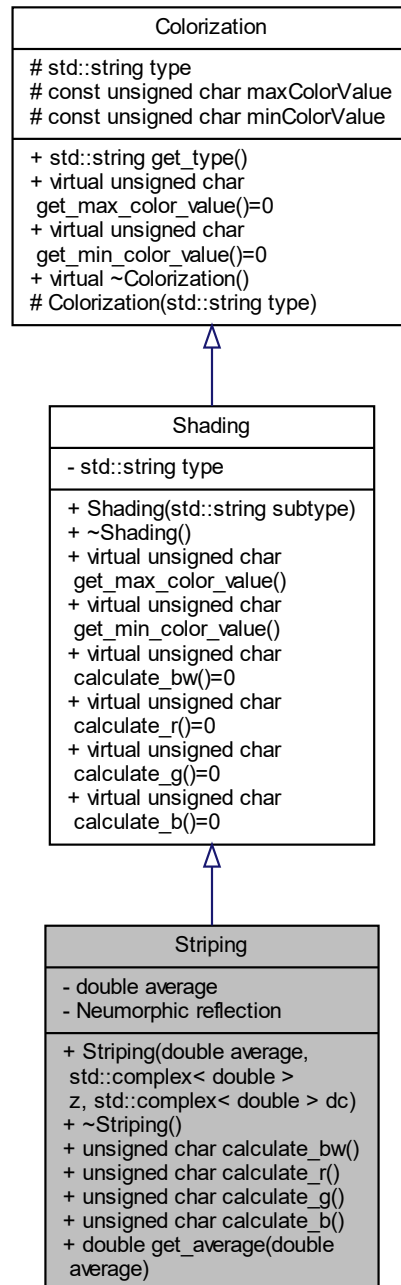
The documentation for this class was generated from the following files:

- D:/Documents/Final/Mandelbrot_Cpp_Final/src/[Shading.h](#)
- D:/Documents/Final/Mandelbrot_Cpp_Final/src/[Shading.cpp](#)

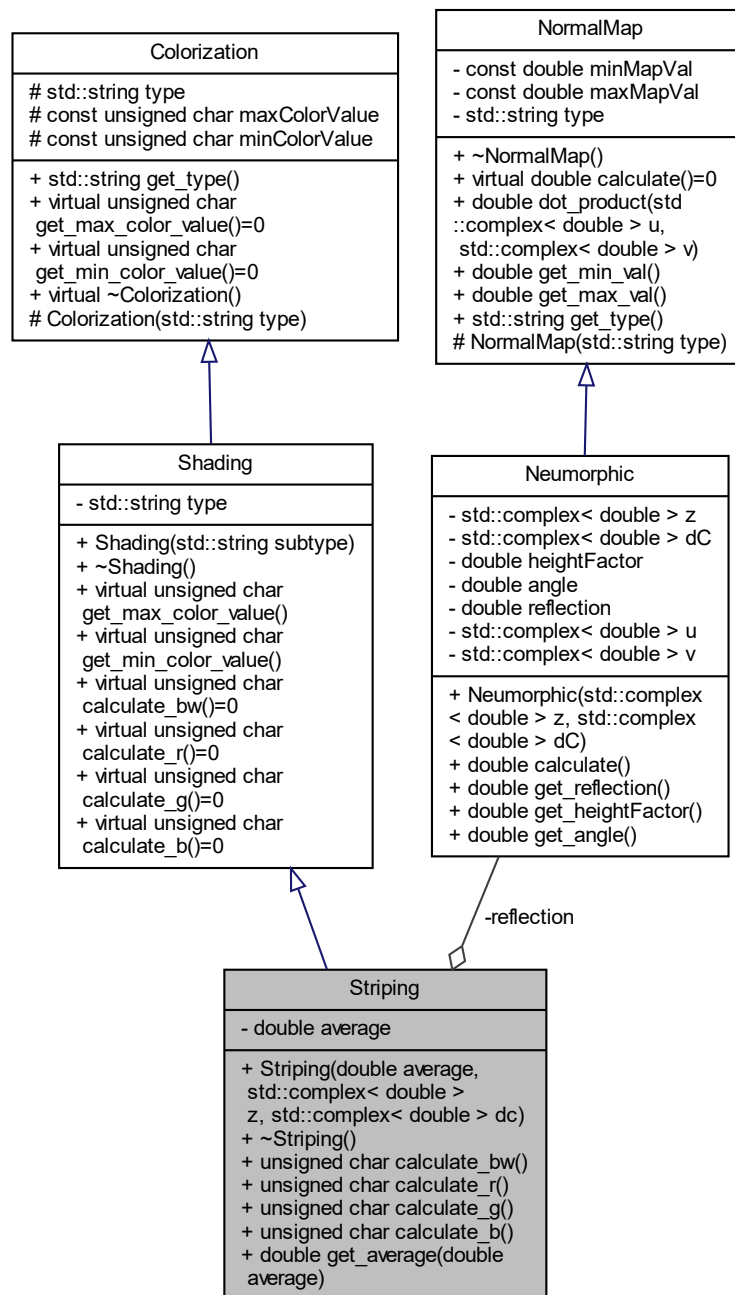
5.9 Striping Class Reference

```
#include <Striping.h>
```

Inheritance diagram for Striping:



Collaboration diagram for Striping:



Public Member Functions

- [Striping](#) (double [average](#), std::complex< double > z, std::complex< double > dc)
- [~Striping](#) ()
- unsigned char [calculate_bw](#) ()
- unsigned char [calculate_r](#) ()
- unsigned char [calculate_g](#) ()

- unsigned char [calculate_b](#) ()
- double [get_average](#) (double [average](#))

Private Attributes

- double [average](#)
- [Neumorphic reflection](#)

Additional Inherited Members

5.9.1 Detailed Description

Definition at line 7 of file [Striping.h](#).

5.9.2 Constructor & Destructor Documentation

5.9.2.1 Striping()

```
Striping::Striping (
    double average,
    std::complex< double > z,
    std::complex< double > dc )
```

Definition at line 5 of file [Striping.cpp](#).

```
00005                                     : Shading("Striping"),
00006 average(average), reflection(z, dc)
00007 {}
```

5.9.2.2 ~Striping()

```
Striping::~Striping ( ) [default]
```

5.9.3 Member Function Documentation

5.9.3.1 calculate_b()

```
unsigned char Striping::calculate_b ( ) [virtual]
```

Implements [Shading](#).

Definition at line 28 of file [Striping.cpp](#).

```
00029 {
00030     return 0;
00031 }
```

5.9.3.2 calculate_bw()

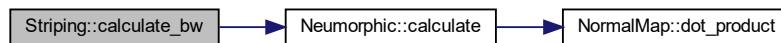
```
unsigned char Striping::calculate_bw ( ) [virtual]
```

Implements [Shading](#).

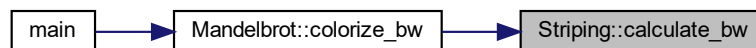
Definition at line 12 of file [Striping.cpp](#).

```
00013 {  
00014     return static_cast<unsigned char>((static_cast<double>((maxColorValue - 1)) - (100.0 *  
00015     average)) * reflection.calculate()); // explicit casting  
00016 }
```

Here is the call graph for this function:



Here is the caller graph for this function:



5.9.3.3 calculate_g()

```
unsigned char Striping::calculate_g ( ) [virtual]
```

Implements [Shading](#).

Definition at line 23 of file [Striping.cpp](#).

```
00024 {  
00025     return 0;  
00026 }
```

5.9.3.4 calculate_r()

```
unsigned char Striping::calculate_r ( ) [virtual]
```

Implements [Shading](#).

Definition at line 18 of file [Striping.cpp](#).

```
00019 {  
00020     return 0;  
00021 }
```

5.9.3.5 get_average()

```
double Striping::get_average (
    double average )
```

Definition at line 33 of file [Striping.cpp](#).

```
00034 {
00035     return average;
00036 }
```

5.9.4 Member Data Documentation

5.9.4.1 average

```
double Striping::average [private]
```

Definition at line 27 of file [Striping.h](#).

5.9.4.2 reflection

```
Neumorphic Striping::reflection [private]
```

Definition at line 29 of file [Striping.h](#).

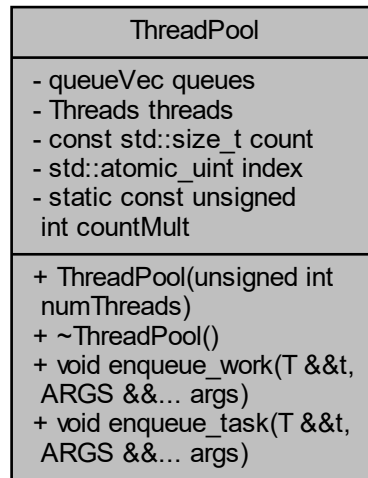
The documentation for this class was generated from the following files:

- D:/Documents/Final/Mandelbrot_Cpp_Final/src/[Striping.h](#)
- D:/Documents/Final/Mandelbrot_Cpp_Final/src/[Striping.cpp](#)

5.10 ThreadPool Class Reference

```
#include <ThreadPool.h>
```


Collaboration diagram for ThreadPool:



Public Member Functions

- [ThreadPool](#) (unsigned int numThreads)
- [~ThreadPool](#) ()
- template<typename T , typename... ARGS>
void [enqueue_work](#) (T &&t, ARGS &&... args)
- template<typename T , typename... ARGS>
void [enqueue_task](#) (T &&t, ARGS &&... args)

Private Types

- using [process](#) = std::function< void(void)>
- using [queue](#) = [UnboundedQueue](#)< [process](#) >
- using [queueVec](#) = std::vector< [queue](#) >
- using [Threads](#) = std::vector< std::thread >

Private Attributes

- [queueVec](#) queues
- [Threads](#) threads
- const std::size_t [count](#)
- std::atomic_uint [index](#) = 0

Static Private Attributes

- static const unsigned int [countMult](#) = 2

5.10.1 Detailed Description

Only using C++ intrinsics

Definition at line 16 of file [ThreadPool.h](#).

5.10.2 Member Typedef Documentation

5.10.2.1 process

```
using ThreadPool::process = std::function<void(void)> [private]
```

Definition at line 70 of file [ThreadPool.h](#).

5.10.2.2 queue

```
using ThreadPool::queue = UnboundedQueue<process> [private]
```

Definition at line 72 of file [ThreadPool.h](#).

5.10.2.3 queueVec

```
using ThreadPool::queueVec = std::vector<queue> [private]
```

Definition at line 74 of file [ThreadPool.h](#).

5.10.2.4 Threads

```
using ThreadPool::Threads = std::vector<std::thread> [private]
```

Definition at line 79 of file [ThreadPool.h](#).

5.10.3 Constructor & Destructor Documentation

5.10.3.1 ThreadPool()

```
ThreadPool::ThreadPool (
    unsigned int numThreads ) [explicit]
```

Definition at line 3 of file [ThreadPool.cpp](#).

```
00003                                     : queues(numThreads), count(numThreads)
00004 {
00005     if (!numThreads)
00006     {
00007         throw std::invalid_argument("thread count must be nonzero!\n");
00008     } else if (numThreads < 0) {
00009         throw std::invalid_argument("thread count must be positive! how did this happen??");
00010     }
00011
00012     auto worker = [this] (auto i) {
00013         while (true)
00014         {
00015             process proc;
00016             for (auto j = 0; j < count * countMult; j++)
00017             {
00018                 if (queues[(i + j) % count].try_pop(proc))
00019                 {
00020                     break;
00021                 }
00022             }
00023             if (!proc && !queues[i].pop(proc))
00024             {
00025                 break;
00026             }
00027             proc();
00028         }
00029     };
00030
00031     threads.reserve(numThreads);
00032
00033     for (auto i = 0; i < numThreads; i++)
00034     {
00035         threads.emplace_back(worker, i);
00036     }
00037 }
```

5.10.3.2 ~ThreadPool()

```
ThreadPool::~~ThreadPool ( )
```

Definition at line 39 of file [ThreadPool.cpp](#).

```
00040 {
00041     for (auto& queue: queues)
00042     {
00043         queue.unblock();
00044     }
00045     for (auto& thread : threads)
00046     {
00047         thread.join();
00048     }
00049 }
```

Here is the call graph for this function:



5.10.4 Member Function Documentation

5.10.4.1 enqueue_task()

```
template<typename T , typename...  ARGS>
void ThreadPool::enqueue_task (
    T && t,
    ARGS &&...  args ) [inline]
```

Definition at line 44 of file [ThreadPool.h](#).

```
00045 {
00046     using taskReturnType = std::invoke_result<T, ARGS...>;
00047     using taskType = std::packaged_task<taskReturnType>;
00048
00049     auto task = std::make_shared<taskType>(std::bind(std::forward<T>(t),
std::forward<ARGS>(args)...));
00050     auto work = [=] () {(*task)();};
00051     auto result = task->get_future();
00052
00053     auto i = index++;
00054
00055     for (auto j = 0; j < count * countMult; j++)
00056     {
00057         if (queues[(i + j) % count].try_push(work))
00058         {
00059             return result;
00060         }
00061     }
00062
00063     queues[i % count].push(std::move(work));
00064
00065     return result;
00066 }
```

5.10.4.2 enqueue_work()

```
template<typename T , typename...  ARGS>
void ThreadPool::enqueue_work (
    T && t,
    ARGS &&...  args ) [inline]
```

Definition at line 25 of file [ThreadPool.h](#).

```
00026 {
00027     auto work = [proc = std::forward<T>(t), tuple = std::make_tuple(std::forward<ARGS>(args)...)]
00028     () {std::apply(proc, tuple);};
00029
00030     auto i = index++;
00031
00032     for (auto j = 0; j < count * countMult; j++)
00033     {
00034         if (queues[(i + j) % count].try_push(work))
00035         {
00036             return;
00037         }
00038     }
00039
00040     queues[i % count].push(std::move(work));
00041 }
```

Here is the caller graph for this function:



5.10.5 Member Data Documentation

5.10.5.1 count

```
const std::size_t ThreadPool::count [private]
```

Definition at line 84 of file [ThreadPool.h](#).

5.10.5.2 countMult

```
const unsigned int ThreadPool::countMult = 2 [inline], [static], [private]
```

Definition at line 88 of file [ThreadPool.h](#).

5.10.5.3 index

```
std::atomic_uint ThreadPool::index = 0 [private]
```

Definition at line 86 of file [ThreadPool.h](#).

5.10.5.4 queues

```
queueVec ThreadPool::queues [private]
```

Definition at line 76 of file [ThreadPool.h](#).

5.10.5.5 threads

`Threads ThreadPool::threads [private]`

Definition at line 81 of file [ThreadPool.h](#).

The documentation for this class was generated from the following files:

- D:/Documents/Final/Mandelbrot_Cpp_Final/src/[ThreadPool.h](#)
- D:/Documents/Final/Mandelbrot_Cpp_Final/src/[ThreadPool.cpp](#)

5.11 UnboundedQueue< T > Class Template Reference

```
#include <UnboundedQueue.h>
```

Collaboration diagram for UnboundedQueue< T >:

UnboundedQueue< T >
<ul style="list-style-type: none"> - queue_t queue - bool is_block - std::mutex queueLock - std::condition_variable condition
<ul style="list-style-type: none"> + UnboundedQueue(bool block=true) + ~UnboundedQueue() + void push(const T &item) + void push(T &&item) + void emplace(ARGS &&... args) + bool try_push(const T &item) + bool try_push(T &&item) + bool pop(T &item) + bool try_pop(T &item) + std::size_t size() const + bool empty() const + void block() + void unblock() + bool blocking() const

Public Member Functions

- [UnboundedQueue](#) (bool [block](#)=true)
- [~UnboundedQueue](#) ()
- void [push](#) (const T &item)
- void [push](#) (T &&item)
- template<typename... ARGS>
void [emplace](#) (ARGS &&... args)
- bool [try_push](#) (const T &item)
- bool [try_push](#) (T &&item)
- bool [pop](#) (T &item)
- bool [try_pop](#) (T &item)
- std::size_t [size](#) () const
- bool [empty](#) () const
- void [block](#) ()
- void [unblock](#) ()
- bool [blocking](#) () const

Private Types

- using [queue_t](#) = std::queue< T >

Private Attributes

- [queue_t](#) [queue](#)
- bool [is_block](#)
- std::mutex [queueLock](#)
- std::condition_variable [condition](#)

5.11.1 Detailed Description

```
template<typename T>
class UnboundedQueue< T >
```

Definition at line 11 of file [UnboundedQueue.h](#).

5.11.2 Member Typedef Documentation

5.11.2.1 queue_t

```
template<typename T >
using UnboundedQueue< T >::queue\_t = std::queue<T> [private]
```

Definition at line 65 of file [UnboundedQueue.h](#).

5.11.3 Constructor & Destructor Documentation

5.11.3.1 UnboundedQueue()

```
template<typename T >
UnboundedQueue< T >::UnboundedQueue (
    bool block = true ) [explicit]
```

Default Parametrized constructor

Parameters

<i>block</i>	
--------------	--

Definition at line 77 of file [UnboundedQueue.h](#).

```
00077                                     : is_block(block)
00078 {}
```

5.11.3.2 ~UnboundedQueue()

```
template<typename T >
UnboundedQueue< T >::~~UnboundedQueue ( ) [default]
```

Default destructor

5.11.4 Member Function Documentation

5.11.4.1 block()

```
template<typename T >
void UnboundedQueue< T >::block
```

Definition at line 183 of file [UnboundedQueue.h](#).

```
00184 {
00185     std::scoped_lock guard(queueLock);
00186     is_block = true;
00187 }
```


5.11.4.2 blocking()

```
template<typename T >
bool UnboundedQueue< T >::blocking
```

Definition at line 200 of file [UnboundedQueue.h](#).

```
00201 {
00202     std::scoped_lock guard(queueLock);
00203     return is_block;
00204 }
```

5.11.4.3 emplace()

```
template<typename T >
template<typename... ARGS>
void UnboundedQueue< T >::emplace (
    ARGS &&... args )
```

Emplace

Template Parameters

<i>Args</i>	
-------------	--

Parameters

<i>args</i>	
-------------	--

Definition at line 102 of file [UnboundedQueue.h](#).

```
00103 {
00104     {
00105         std::scoped_lock guard(queueLock);
00106         queue.emplace(std::forward<>(args)...);
00107     }
00108     condition.notify_one();
00109 }
```

5.11.4.4 empty()

```
template<typename T >
bool UnboundedQueue< T >::empty
```

Definition at line 176 of file [UnboundedQueue.h](#).

```
00177 {
00178     std::scoped_lock guard(queueLock);
00179     return queue.empty();
00180 }
```

5.11.4.5 pop()

```
template<typename T >
bool UnboundedQueue< T >::pop (
    T & item )
```

Definition at line 142 of file [UnboundedQueue.h](#).

```
00143 {
00144     std::unique_lock guard(queueLock);
00145     condition.wait(guard, [&] () {return !queue.empty() || !is_block;});
00146     if (queue.empty())
00147     {
00148         return false;
00149     }
00150     item = std::move(queue.front());
00151     queue.pop();
00152     return true;
00153 }
```

5.11.4.6 push() [1/2]

```
template<typename T >
void UnboundedQueue< T >::push (
    const T & item )
```

Parameters

<i>item</i>	- const reference
-------------	-------------------

Definition at line 81 of file [UnboundedQueue.h](#).

```
00082 {
00083     {
00084         std::scoped_lock guard(queueLock);
00085         queue.push(item);
00086     }
00087     condition.notify_one();
00088 }
```

5.11.4.7 push() [2/2]

```
template<typename T >
void UnboundedQueue< T >::push (
    T && item )
```

Parameters

<i>item</i>	- double reference
-------------	--------------------

Definition at line 91 of file [UnboundedQueue.h](#).

```
00092 {
00093     {
00094         std::scoped_lock guard(queueLock);
00095         queue.push(std::move(item));
00096     }
00097     condition.notify_one();
```

```
00098 }
```

5.11.4.8 size()

```
template<typename T >
std::size_t UnboundedQueue< T >::size
```

Definition at line 169 of file [UnboundedQueue.h](#).

```
00170 {
00171     std::scoped_lock guard(queueLock);
00172     return queue.size();
00173 }
```

5.11.4.9 try_pop()

```
template<typename T >
bool UnboundedQueue< T >::try_pop (
    T & item )
```

Definition at line 156 of file [UnboundedQueue.h](#).

```
00157 {
00158     std::unique_lock guard(queueLock, std::try_to_lock);
00159     if (!guard || queue.empty())
00160     {
00161         return false;
00162     }
00163     item = std::move(queue.front());
00164     queue.pop();
00165     return true;
00166 }
```

5.11.4.10 try_push() [1/2]

```
template<typename T >
bool UnboundedQueue< T >::try_push (
    const T & item )
```

Definition at line 112 of file [UnboundedQueue.h](#).

```
00113 {
00114     {
00115         std::unique_lock guard(queueLock, std::try_to_lock);
00116         if (!guard)
00117         {
00118             return false;
00119         }
00120         queue.push(item);
00121     }
00122     condition.notify_one();
00123     return true;
00124 }
```

5.11.4.11 try_push() [2/2]

```
template<typename T >
bool UnboundedQueue< T >::try_push (
    T && item )
```

Definition at line 127 of file [UnboundedQueue.h](#).

```
00128 {
00129     {
00130         std::unique_lock guard(queueLock, std::try_to_lock);
00131         if (!guard)
00132         {
00133             return false;
00134         }
00135         queue.push(std::move(item));
00136     }
00137     condition.notify_one();
00138     return true;
00139 }
```

5.11.4.12 unblock()

```
template<typename T >
void UnboundedQueue< T >::unblock
```

Definition at line 190 of file [UnboundedQueue.h](#).

```
00191 {
00192     {
00193         std::scoped_lock guard(queueLock);
00194         is_block = false;
00195     }
00196     condition.notify_all();
00197 }
```

Here is the caller graph for this function:



5.11.5 Member Data Documentation

5.11.5.1 condition

```
template<typename T >
std::condition_variable UnboundedQueue< T >::condition [private]
```

Definition at line 72 of file [UnboundedQueue.h](#).

5.11.5.2 is_block

```
template<typename T >  
bool UnboundedQueue< T >::is_block [private]
```

Definition at line 68 of file [UnboundedQueue.h](#).

5.11.5.3 queue

```
template<typename T >  
queue_t UnboundedQueue< T >::queue [private]
```

Definition at line 66 of file [UnboundedQueue.h](#).

5.11.5.4 queueLock

```
template<typename T >  
std::mutex UnboundedQueue< T >::queueLock [mutable], [private]
```

Definition at line 70 of file [UnboundedQueue.h](#).

The documentation for this class was generated from the following file:

- D:/Documents/Final/Mandelbrot_Cpp_Final/src/[UnboundedQueue.h](#)

Chapter 6

File Documentation

6.1 D:/Documents/Final/Mandelbrot_Cpp_Final/cmake-build-debug/↵ CMakeFiles/3.21.1/CompilerIdC/CMakeCCompilerId.c File Reference

Macros

- `#define __has_include(x) 0`
- `#define COMPILER_ID ""`
- `#define STRINGIFY_HELPER(X) #X`
- `#define STRINGIFY(X) STRINGIFY_HELPER(X)`
- `#define PLATFORM_ID`
- `#define ARCHITECTURE_ID`
- `#define DEC(n)`
- `#define HEX(n)`
- `#define C_DIALECT`

Functions

- `int main (int argc, char *argv[])`

Variables

- `char const * info_compiler = "INFO" ":" "compiler[" COMPILER_ID "]"`
- `char const * info_platform = "INFO" ":" "platform[" PLATFORM_ID "]"`
- `char const * info_arch = "INFO" ":" "arch[" ARCHITECTURE_ID "]"`
- `const char * info_language_dialect_default`

6.1.1 Macro Definition Documentation

6.1.1.1 `__has_include`

```
#define __has_include(  
    x ) 0
```

Definition at line 17 of file [CMakeCCompilerId.c](#).

6.1.1.2 `ARCHITECTURE_ID`

```
#define ARCHITECTURE_ID
```

Definition at line 668 of file [CMakeCCompilerId.c](#).

6.1.1.3 `C_DIALECT`

```
#define C_DIALECT
```

Definition at line 757 of file [CMakeCCompilerId.c](#).

6.1.1.4 `COMPILER_ID`

```
#define COMPILER_ID ""
```

Definition at line 412 of file [CMakeCCompilerId.c](#).

6.1.1.5 `DEC`

```
#define DEC(  
    n )
```

Value:

```
('0' + ((n) / 10000000) % 10), \
('0' + ((n) / 1000000) % 10), \
('0' + ((n) / 100000) % 10), \
('0' + ((n) / 10000) % 10), \
('0' + ((n) / 1000) % 10), \
('0' + ((n) / 100) % 10), \
('0' + ((n) / 10) % 10), \
('0' + ((n) % 10))
```

Definition at line 672 of file [CMakeCCompilerId.c](#).

6.1.1.6 HEX

```
#define HEX(  
    n )
```

Value:

```
('0' + ((n)>>28 & 0xF)), \  
( '0' + ((n)>>24 & 0xF)), \  
( '0' + ((n)>>20 & 0xF)), \  
( '0' + ((n)>>16 & 0xF)), \  
( '0' + ((n)>>12 & 0xF)), \  
( '0' + ((n)>>8  & 0xF)), \  
( '0' + ((n)>>4  & 0xF)), \  
( '0' + ((n)    & 0xF))
```

Definition at line 683 of file [CMakeCCompilerId.c](#).

6.1.1.7 PLATFORM_ID

```
#define PLATFORM_ID
```

Definition at line 540 of file [CMakeCCompilerId.c](#).

6.1.1.8 STRINGIFY

```
#define STRINGIFY(  
    X ) STRINGIFY_HELPER(X)
```

Definition at line 433 of file [CMakeCCompilerId.c](#).

6.1.1.9 STRINGIFY_HELPER

```
#define STRINGIFY_HELPER(  
    X ) #X
```

Definition at line 432 of file [CMakeCCompilerId.c](#).

6.1.2 Function Documentation

6.1.2.1 main()

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

Definition at line 781 of file [CMakeCCompilerId.c](#).

```
00783 {
00784     int require = 0;
00785     require += info_compiler[argc];
00786     require += info_platform[argc];
00787     require += info_arch[argc];
00788     #ifdef COMPILER_VERSION_MAJOR
00789     require += info_version[argc];
00790 #endif
00791     #ifdef COMPILER_VERSION_INTERNAL
00792     require += info_version_internal[argc];
00793 #endif
00794     #ifdef SIMULATE_ID
00795     require += info_simulate[argc];
00796 #endif
00797     #ifdef SIMULATE_VERSION_MAJOR
00798     require += info_simulate_version[argc];
00799 #endif
00800     #if defined(__CRAYXT_COMPUTE_LINUX_TARGET)
00801     require += info_cray[argc];
00802 #endif
00803     require += info_language_dialect_default[argc];
00804     (void)argv;
00805     return require;
00806 }
```

6.1.3 Variable Documentation

6.1.3.1 info_arch

```
char const* info_arch = "INFO" ":" "arch[" ARCHITECTURE_ID "]"
```

Definition at line 749 of file [CMakeCCompilerId.c](#).

6.1.3.2 info_compiler

```
char const* info_compiler = "INFO" ":" "compiler[" COMPILER_ID "]"
```

Definition at line 419 of file [CMakeCCompilerId.c](#).

6.1.3.3 info_language_dialect_default

```
const char* info_language_dialect_default
```

Initial value:

```
=
"INFO" ":" "dialect_default[" C_DIALECT "]"
```

Definition at line 770 of file [CMakeCCompilerId.c](#).

6.1.3.4 info_platform

```
char const* info_platform = "INFO" ":" "platform[" PLATFORM_ID "]"
```

Definition at line 748 of file [CMakeCCompilerId.c](#).

6.2 CMakeCCompilerId.c

[Go to the documentation of this file.](#)

```
00001 #ifdef __cplusplus
00002 # error "A C++ compiler has been selected for C."
00003 #endif
00004
00005 #if defined(__18CXX)
00006 # define ID_VOID_MAIN
00007 #endif
00008 #if defined(__CLASSIC_C__)
00009 /* cv-qualifiers did not exist in K&R C */
00010 # define const
00011 # define volatile
00012 #endif
00013
00014 #if !defined(__has_include)
00015 /* If the compiler does not have __has_include, pretend the answer is
00016    always no. */
00017 # define __has_include(x) 0
00018 #endif
00019
00020
00021 /* Version number components: V=Version, R=Revision, P=Patch
00022    Version date components: YYYY=Year, MM=Month, DD=Day */
00023
00024 #if defined(__INTEL_COMPILER) || defined(__ICC)
00025 # define COMPILER_ID "Intel"
00026 # if defined(_MSC_VER)
00027 #   define SIMULATE_ID "MSVC"
00028 # endif
00029 # if defined(__GNUC__)
00030 #   define SIMULATE_ID "GNU"
00031 # endif
00032 /* __INTEL_COMPILER = VRP prior to 2021, and then VVVV for 2021 and later,
00033    except that a few beta releases use the old format with V=2021. */
00034 # if __INTEL_COMPILER < 2021 || __INTEL_COMPILER == 202110 || __INTEL_COMPILER == 202111
00035 #   define COMPILER_VERSION_MAJOR DEC(__INTEL_COMPILER/100)
00036 #   define COMPILER_VERSION_MINOR DEC(__INTEL_COMPILER/10 % 10)
00037 #   if defined(__INTEL_COMPILER_UPDATE)
00038 #     define COMPILER_VERSION_PATCH DEC(__INTEL_COMPILER_UPDATE)
00039 #   else
00040 #     define COMPILER_VERSION_PATCH DEC(__INTEL_COMPILER % 10)
00041 #   endif
00042 # else
00043 #   define COMPILER_VERSION_MAJOR DEC(__INTEL_COMPILER)
00044 #   define COMPILER_VERSION_MINOR DEC(__INTEL_COMPILER_UPDATE)
00045 /* The third version component from --version is an update index,
00046    but no macro is provided for it. */
00047 #   define COMPILER_VERSION_PATCH DEC(0)
00048 # endif
00049 # if defined(__INTEL_COMPILER_BUILD_DATE)
00050 /* __INTEL_COMPILER_BUILD_DATE = YYYYMMDD */
00051 #   define COMPILER_VERSION_TWEAK DEC(__INTEL_COMPILER_BUILD_DATE)
00052 # endif
00053 # if defined(_MSC_VER)
00054 /* _MSC_VER = VVRR */
00055 #   define SIMULATE_VERSION_MAJOR DEC(_MSC_VER / 100)
00056 #   define SIMULATE_VERSION_MINOR DEC(_MSC_VER % 100)
00057 # endif
00058 # if defined(__GNUC__)
00059 #   define SIMULATE_VERSION_MAJOR DEC(__GNUC__)
00060 # elif defined(__GNUG__)
00061 #   define SIMULATE_VERSION_MAJOR DEC(__GNUG__)
00062 # endif
00063 # if defined(__GNUC_MINOR__)
00064 #   define SIMULATE_VERSION_MINOR DEC(__GNUC_MINOR__)
00065 # endif
00066 # if defined(__GNUC_PATCHLEVEL__)
00067 #   define SIMULATE_VERSION_PATCH DEC(__GNUC_PATCHLEVEL__)
00068 # endif
00069
```

```

00070 #elif (defined(__clang__) && defined(__INTEL_CLANG_COMPILER)) || defined(__INTEL_LLVM_COMPILER)
00071 # define COMPILER_ID "IntelLLVM"
00072 #if defined(_MSC_VER)
00073 # define SIMULATE_ID "MSVC"
00074 #endif
00075 #if defined(__GNUC__)
00076 # define SIMULATE_ID "GNU"
00077 #endif
00078 /* __INTEL_LLVM_COMPILER = VVVVRP prior to 2021.2.0, VVVVRRPP for 2021.2.0 and
00079  * later. Look for 6 digit vs. 8 digit version number to decide encoding.
00080  * VVVV is no smaller than the current year when a version is released.
00081  */
00082 #if __INTEL_LLVM_COMPILER < 1000000L
00083 # define COMPILER_VERSION_MAJOR DEC(__INTEL_LLVM_COMPILER/100)
00084 # define COMPILER_VERSION_MINOR DEC(__INTEL_LLVM_COMPILER/10 % 10)
00085 # define COMPILER_VERSION_PATCH DEC(__INTEL_LLVM_COMPILER % 10)
00086 #else
00087 # define COMPILER_VERSION_MAJOR DEC(__INTEL_LLVM_COMPILER/10000)
00088 # define COMPILER_VERSION_MINOR DEC(__INTEL_LLVM_COMPILER/100 % 100)
00089 # define COMPILER_VERSION_PATCH DEC(__INTEL_LLVM_COMPILER % 100)
00090 #endif
00091 #if defined(_MSC_VER)
00092 /* _MSC_VER = VVRR */
00093 # define SIMULATE_VERSION_MAJOR DEC(_MSC_VER / 100)
00094 # define SIMULATE_VERSION_MINOR DEC(_MSC_VER % 100)
00095 #endif
00096 #if defined(__GNUC__)
00097 # define SIMULATE_VERSION_MAJOR DEC(__GNUC__)
00098 #elif defined(__GNUG__)
00099 # define SIMULATE_VERSION_MAJOR DEC(__GNUG__)
00100 #endif
00101 #if defined(__GNUC_MINOR__)
00102 # define SIMULATE_VERSION_MINOR DEC(__GNUC_MINOR__)
00103 #endif
00104 #if defined(__GNUC_PATCHLEVEL__)
00105 # define SIMULATE_VERSION_PATCH DEC(__GNUC_PATCHLEVEL__)
00106 #endif
00107
00108 #elif defined(__PATHCC__)
00109 # define COMPILER_ID "PathScale"
00110 # define COMPILER_VERSION_MAJOR DEC(__PATHCC__)
00111 # define COMPILER_VERSION_MINOR DEC(__PATHCC_MINOR__)
00112 # if defined(__PATHCC_PATCHLEVEL__)
00113 # define COMPILER_VERSION_PATCH DEC(__PATHCC_PATCHLEVEL__)
00114 # endif
00115
00116 #elif defined(__BORLANDC__) && defined(__CODEGEARC_VERSION__)
00117 # define COMPILER_ID "Embarcadero"
00118 # define COMPILER_VERSION_MAJOR HEX(__CODEGEARC_VERSION__>24 & 0x00FF)
00119 # define COMPILER_VERSION_MINOR HEX(__CODEGEARC_VERSION__>16 & 0x00FF)
00120 # define COMPILER_VERSION_PATCH DEC(__CODEGEARC_VERSION__ & 0xFFFF)
00121
00122 #elif defined(__BORLANDC__)
00123 # define COMPILER_ID "Borland"
00124 /* __BORLANDC__ = 0xVRR */
00125 # define COMPILER_VERSION_MAJOR HEX(__BORLANDC__>8)
00126 # define COMPILER_VERSION_MINOR HEX(__BORLANDC__ & 0xFF)
00127
00128 #elif defined(__WATCOMC__) && __WATCOMC__ < 1200
00129 # define COMPILER_ID "Watcom"
00130 /* __WATCOMC__ = VVRR */
00131 # define COMPILER_VERSION_MAJOR DEC(__WATCOMC__ / 100)
00132 # define COMPILER_VERSION_MINOR DEC((__WATCOMC__ / 10) % 10)
00133 # if (__WATCOMC__ % 10) > 0
00134 # define COMPILER_VERSION_PATCH DEC(__WATCOMC__ % 10)
00135 # endif
00136
00137 #elif defined(__WATCOMC__)
00138 # define COMPILER_ID "OpenWatcom"
00139 /* __WATCOMC__ = VVRP + 1100 */
00140 # define COMPILER_VERSION_MAJOR DEC((__WATCOMC__ - 1100) / 100)
00141 # define COMPILER_VERSION_MINOR DEC((__WATCOMC__ / 10) % 10)
00142 # if (__WATCOMC__ % 10) > 0
00143 # define COMPILER_VERSION_PATCH DEC(__WATCOMC__ % 10)
00144 # endif
00145
00146 #elif defined(__SUNPRO_C)
00147 # define COMPILER_ID "SunPro"
00148 # if __SUNPRO_C >= 0x5100
00149 /* __SUNPRO_C = 0xVRRP */
00150 # define COMPILER_VERSION_MAJOR HEX(__SUNPRO_C>12)
00151 # define COMPILER_VERSION_MINOR HEX(__SUNPRO_C>4 & 0xFF)
00152 # define COMPILER_VERSION_PATCH HEX(__SUNPRO_C & 0xF)
00153 # else
00154 /* __SUNPRO_CC = 0xVRP */
00155 # define COMPILER_VERSION_MAJOR HEX(__SUNPRO_C>8)
00156 # define COMPILER_VERSION_MINOR HEX(__SUNPRO_C>4 & 0xF)

```

```

00157 # define COMPILER_VERSION_PATCH HEX(__SUNPRO_C    & 0xF)
00158 # endif
00159
00160 #elif defined(__HP_cc)
00161 # define COMPILER_ID "HP"
00162 /* __HP_cc = VVRRPP */
00163 # define COMPILER_VERSION_MAJOR DEC(__HP_cc/10000)
00164 # define COMPILER_VERSION_MINOR DEC(__HP_cc/100 % 100)
00165 # define COMPILER_VERSION_PATCH DEC(__HP_cc    % 100)
00166
00167 #elif defined(__DECC)
00168 # define COMPILER_ID "Compaq"
00169 /* __DECC_VER = VVVRTPPPP */
00170 # define COMPILER_VERSION_MAJOR DEC(__DECC_VER/10000000)
00171 # define COMPILER_VERSION_MINOR DEC(__DECC_VER/100000 % 100)
00172 # define COMPILER_VERSION_PATCH DEC(__DECC_VER    % 10000)
00173
00174 #elif defined(__IBMC__) && defined(__COMPILER_VER__)
00175 # define COMPILER_ID "zOS"
00176 /* __IBMC__ = VRP */
00177 # define COMPILER_VERSION_MAJOR DEC(__IBMC__/100)
00178 # define COMPILER_VERSION_MINOR DEC(__IBMC__/10 % 10)
00179 # define COMPILER_VERSION_PATCH DEC(__IBMC__    % 10)
00180
00181 #elif defined(__ibmxl__) && defined(__clang__)
00182 # define COMPILER_ID "XLClang"
00183 # define COMPILER_VERSION_MAJOR DEC(__ibmxl_version__)
00184 # define COMPILER_VERSION_MINOR DEC(__ibmxl_release__)
00185 # define COMPILER_VERSION_PATCH DEC(__ibmxl_modification__)
00186 # define COMPILER_VERSION_TWEAK DEC(__ibmxl_ptf_fix_level__)
00187
00188
00189 #elif defined(__IBMC__) && !defined(__COMPILER_VER__) && __IBMC__ >= 800
00190 # define COMPILER_ID "XL"
00191 /* __IBMC__ = VRP */
00192 # define COMPILER_VERSION_MAJOR DEC(__IBMC__/100)
00193 # define COMPILER_VERSION_MINOR DEC(__IBMC__/10 % 10)
00194 # define COMPILER_VERSION_PATCH DEC(__IBMC__    % 10)
00195
00196 #elif defined(__IBMC__) && !defined(__COMPILER_VER__) && __IBMC__ < 800
00197 # define COMPILER_ID "VisualAge"
00198 /* __IBMC__ = VRP */
00199 # define COMPILER_VERSION_MAJOR DEC(__IBMC__/100)
00200 # define COMPILER_VERSION_MINOR DEC(__IBMC__/10 % 10)
00201 # define COMPILER_VERSION_PATCH DEC(__IBMC__    % 10)
00202
00203 #elif defined(__NVCOMPILER)
00204 # define COMPILER_ID "NVHPC"
00205 # define COMPILER_VERSION_MAJOR DEC(__NVCOMPILER_MAJOR__)
00206 # define COMPILER_VERSION_MINOR DEC(__NVCOMPILER_MINOR__)
00207 # if defined(__NVCOMPILER_PATCHLEVEL__)
00208 #   define COMPILER_VERSION_PATCH DEC(__NVCOMPILER_PATCHLEVEL__)
00209 # endif
00210
00211 #elif defined(__PGI)
00212 # define COMPILER_ID "PGI"
00213 # define COMPILER_VERSION_MAJOR DEC(__PGIC__)
00214 # define COMPILER_VERSION_MINOR DEC(__PGIC_MINOR__)
00215 # if defined(__PGIC_PATCHLEVEL__)
00216 #   define COMPILER_VERSION_PATCH DEC(__PGIC_PATCHLEVEL__)
00217 # endif
00218
00219 #elif defined(__CRAYC)
00220 # define COMPILER_ID "Cray"
00221 # define COMPILER_VERSION_MAJOR DEC(_RELEASE_MAJOR)
00222 # define COMPILER_VERSION_MINOR DEC(_RELEASE_MINOR)
00223
00224 #elif defined(__TI_COMPILER_VERSION__)
00225 # define COMPILER_ID "TI"
00226 /* __TI_COMPILER_VERSION__ = VVRRRRPPPP */
00227 # define COMPILER_VERSION_MAJOR DEC(__TI_COMPILER_VERSION__/1000000)
00228 # define COMPILER_VERSION_MINOR DEC(__TI_COMPILER_VERSION__/1000 % 1000)
00229 # define COMPILER_VERSION_PATCH DEC(__TI_COMPILER_VERSION__    % 1000)
00230
00231 #elif defined(__CLANG_FUJITSU)
00232 # define COMPILER_ID "FujitsuClang"
00233 # define COMPILER_VERSION_MAJOR DEC(__FCC_major__)
00234 # define COMPILER_VERSION_MINOR DEC(__FCC_minor__)
00235 # define COMPILER_VERSION_PATCH DEC(__FCC_patchlevel__)
00236 # define COMPILER_VERSION_INTERNAL_STR __clang_version__
00237
00238
00239 #elif defined(__FUJITSU)
00240 # define COMPILER_ID "Fujitsu"
00241 # if defined(__FCC_version__)
00242 #   define COMPILER_VERSION __FCC_version__
00243 # elif defined(__FCC_major__)

```

```

00244 #   define COMPILER_VERSION_MAJOR DEC(__FCC_major__)
00245 #   define COMPILER_VERSION_MINOR DEC(__FCC_minor__)
00246 #   define COMPILER_VERSION_PATCH DEC(__FCC_patchlevel__)
00247 # endif
00248 # if defined(__fcc_version)
00249 #   define COMPILER_VERSION_INTERNAL DEC(__fcc_version)
00250 # elif defined(__FCC_VERSION)
00251 #   define COMPILER_VERSION_INTERNAL DEC(__FCC_VERSION)
00252 # endif
00253
00254
00255 #elif defined(__ghs__)
00256 # define COMPILER_ID "GHS"
00257 /* __GHS_VERSION_NUMBER = VVVVRP */
00258 # ifdef __GHS_VERSION_NUMBER
00259 #   define COMPILER_VERSION_MAJOR DEC(__GHS_VERSION_NUMBER / 100)
00260 #   define COMPILER_VERSION_MINOR DEC(__GHS_VERSION_NUMBER / 10 % 10)
00261 #   define COMPILER_VERSION_PATCH DEC(__GHS_VERSION_NUMBER % 10)
00262 # endif
00263
00264 #elif defined(__TINYC__)
00265 # define COMPILER_ID "TinyCC"
00266
00267 #elif defined(__BCC__)
00268 # define COMPILER_ID "Bruce"
00269
00270 #elif defined(__SCO_VERSION__)
00271 # define COMPILER_ID "SCO"
00272
00273 #elif defined(__ARMCC_VERSION) && !defined(__clang__)
00274 # define COMPILER_ID "ARMCC"
00275 #if __ARMCC_VERSION >= 1000000
00276 /* __ARMCC_VERSION = VVRPPPP */
00277 #   define COMPILER_VERSION_MAJOR DEC(__ARMCC_VERSION/1000000)
00278 #   define COMPILER_VERSION_MINOR DEC(__ARMCC_VERSION/10000 % 100)
00279 #   define COMPILER_VERSION_PATCH DEC(__ARMCC_VERSION % 10000)
00280 #else
00281 /* __ARMCC_VERSION = VRPPPP */
00282 #   define COMPILER_VERSION_MAJOR DEC(__ARMCC_VERSION/100000)
00283 #   define COMPILER_VERSION_MINOR DEC(__ARMCC_VERSION/10000 % 10)
00284 #   define COMPILER_VERSION_PATCH DEC(__ARMCC_VERSION % 10000)
00285 #endif
00286
00287
00288 #elif defined(__clang__) && defined(__apple_build_version__)
00289 # define COMPILER_ID "AppleClang"
00290 # if defined(_MSC_VER)
00291 #   define SIMULATE_ID "MSVC"
00292 # endif
00293 # define COMPILER_VERSION_MAJOR DEC(__clang_major__)
00294 # define COMPILER_VERSION_MINOR DEC(__clang_minor__)
00295 # define COMPILER_VERSION_PATCH DEC(__clang_patchlevel__)
00296 # if defined(_MSC_VER)
00297 /* _MSC_VER = VVRR */
00298 #   define SIMULATE_VERSION_MAJOR DEC(_MSC_VER / 100)
00299 #   define SIMULATE_VERSION_MINOR DEC(_MSC_VER % 100)
00300 # endif
00301 # define COMPILER_VERSION_TWEAK DEC(__apple_build_version__)
00302
00303 #elif defined(__clang__) && defined(__ARMCOMPILER_VERSION)
00304 # define COMPILER_ID "ARMClang"
00305 #   define COMPILER_VERSION_MAJOR DEC(__ARMCOMPILER_VERSION/1000000)
00306 #   define COMPILER_VERSION_MINOR DEC(__ARMCOMPILER_VERSION/10000 % 100)
00307 #   define COMPILER_VERSION_PATCH DEC(__ARMCOMPILER_VERSION % 10000)
00308 #   define COMPILER_VERSION_INTERNAL DEC(__ARMCOMPILER_VERSION)
00309
00310 #elif defined(__clang__) && __has_include(<hip/hip_version.h>)
00311 # define COMPILER_ID "ROCMClang"
00312 # if defined(_MSC_VER)
00313 #   define SIMULATE_ID "MSVC"
00314 # elif defined(__clang__)
00315 #   define SIMULATE_ID "Clang"
00316 # elif defined(__GNUC__)
00317 #   define SIMULATE_ID "GNU"
00318 # endif
00319 # if defined(__clang__) && __has_include(<hip/hip_version.h>)
00320 #   include <hip/hip_version.h>
00321 #   define COMPILER_VERSION_MAJOR DEC(HIP_VERSION_MAJOR)
00322 #   define COMPILER_VERSION_MINOR DEC(HIP_VERSION_MINOR)
00323 #   define COMPILER_VERSION_PATCH DEC(HIP_VERSION_PATCH)
00324 # endif
00325
00326 #elif defined(__clang__)
00327 # define COMPILER_ID "Clang"
00328 # if defined(_MSC_VER)
00329 #   define SIMULATE_ID "MSVC"
00330 # endif

```

```

00331 # define COMPILER_VERSION_MAJOR DEC(__clang_major__)
00332 # define COMPILER_VERSION_MINOR DEC(__clang_minor__)
00333 # define COMPILER_VERSION_PATCH DEC(__clang_patchlevel__)
00334 # if defined(_MSC_VER)
00335     /* _MSC_VER = VVRR */
00336 #   define SIMULATE_VERSION_MAJOR DEC(_MSC_VER / 100)
00337 #   define SIMULATE_VERSION_MINOR DEC(_MSC_VER % 100)
00338 # endif
00339
00340 #elif defined(__GNUC__)
00341 #   define COMPILER_ID "GNU"
00342 #   define COMPILER_VERSION_MAJOR DEC(__GNUC__)
00343 #   if defined(__GNUC_MINOR__)
00344 #       define COMPILER_VERSION_MINOR DEC(__GNUC_MINOR__)
00345 #   endif
00346 #   if defined(__GNUC_PATCHLEVEL__)
00347 #       define COMPILER_VERSION_PATCH DEC(__GNUC_PATCHLEVEL__)
00348 #   endif
00349
00350 #elif defined(_MSC_VER)
00351 #   define COMPILER_ID "MSVC"
00352     /* _MSC_VER = VVRR */
00353 #   define COMPILER_VERSION_MAJOR DEC(_MSC_VER / 100)
00354 #   define COMPILER_VERSION_MINOR DEC(_MSC_VER % 100)
00355 #   if defined(_MSC_FULL_VER)
00356 #       if _MSC_FULL_VER >= 1400
00357         /* _MSC_FULL_VER = VVRRPPPP */
00358 #         define COMPILER_VERSION_PATCH DEC(_MSC_FULL_VER % 100000)
00359 #       else
00360         /* _MSC_FULL_VER = VVRRPPPP */
00361 #         define COMPILER_VERSION_PATCH DEC(_MSC_FULL_VER % 10000)
00362 #       endif
00363 #   endif
00364 #   if defined(_MSC_BUILD)
00365 #       define COMPILER_VERSION_TWEAK DEC(_MSC_BUILD)
00366 #   endif
00367
00368 #elif defined(__VISUALDSPVERSION__) || defined(__ADSPBLACKFIN__) || defined(__ADSPTS__) ||
    defined(__ADSP21000__)
00369 #   define COMPILER_ID "ADSP"
00370 #   if defined(__VISUALDSPVERSION__)
00371     /* __VISUALDSPVERSION__ = 0xVVRRPP00 */
00372 #   define COMPILER_VERSION_MAJOR HEX(__VISUALDSPVERSION__>>24)
00373 #   define COMPILER_VERSION_MINOR HEX(__VISUALDSPVERSION__>>16 & 0xFF)
00374 #   define COMPILER_VERSION_PATCH HEX(__VISUALDSPVERSION__>>8 & 0xFF)
00375 #   endif
00376
00377 #elif defined(__IAR_SYSTEMS_ICC__) || defined(__IAR_SYSTEMS_ICC)
00378 #   define COMPILER_ID "IAR"
00379 #   if defined(__VER__) && defined(__ICCARM__)
00380 #       define COMPILER_VERSION_MAJOR DEC((__VER__) / 1000000)
00381 #       define COMPILER_VERSION_MINOR DEC(((__VER__) / 1000) % 1000)
00382 #       define COMPILER_VERSION_PATCH DEC((__VER__) % 1000)
00383 #       define COMPILER_VERSION_INTERNAL DEC(__IAR_SYSTEMS_ICC__)
00384 #   elif defined(__VER__) && (defined(__ICCAVR__) || defined(__ICCRX__) || defined(__ICCRH850__) ||
        defined(__ICCRL78__) || defined(__ICC430__) || defined(__ICCRISCV__) || defined(__ICCV850__) ||
        defined(__ICC8051__) || defined(__ICCSTM8__))
00385 #       define COMPILER_VERSION_MAJOR DEC((__VER__) / 100)
00386 #       define COMPILER_VERSION_MINOR DEC((__VER__) - (((__VER__) / 100)*100))
00387 #       define COMPILER_VERSION_PATCH DEC(__SUBVERSION__)
00388 #       define COMPILER_VERSION_INTERNAL DEC(__IAR_SYSTEMS_ICC__)
00389 #   endif
00390
00391 #elif defined(__SDCC_VERSION_MAJOR) || defined(SDCC)
00392 #   define COMPILER_ID "SDCC"
00393 #   if defined(__SDCC_VERSION_MAJOR)
00394 #       define COMPILER_VERSION_MAJOR DEC(__SDCC_VERSION_MAJOR)
00395 #       define COMPILER_VERSION_MINOR DEC(__SDCC_VERSION_MINOR)
00396 #       define COMPILER_VERSION_PATCH DEC(__SDCC_VERSION_PATCH)
00397 #   else
00398     /* SDCC = VRP */
00399 #       define COMPILER_VERSION_MAJOR DEC(SDCC/100)
00400 #       define COMPILER_VERSION_MINOR DEC(SDCC/10 % 10)
00401 #       define COMPILER_VERSION_PATCH DEC(SDCC % 10)
00402 #   endif
00403
00404
00405 /* These compilers are either not known or too old to define an
00406 identification macro. Try to identify the platform and guess that
00407 it is the native compiler. */
00408 #elif defined(__hpux) || defined(__hpua)
00409 #   define COMPILER_ID "HP"
00410
00411 #else /* unknown compiler */
00412 #   define COMPILER_ID ""
00413 #endif
00414

```

```

00415 /* Construct the string literal in pieces to prevent the source from
00416      getting matched. Store it in a pointer rather than an array
00417      because some compilers will just produce instructions to fill the
00418      array rather than assigning a pointer to a static array. */
00419 char const* info_compiler = "INFO" ":" "compiler[" COMPILER_ID "];"
00420 #ifdef SIMULATE_ID
00421 char const* info_simulate = "INFO" ":" "simulate[" SIMULATE_ID "];"
00422 #endif
00423
00424 #ifdef __QNXNTO__
00425 char const* qnxnto = "INFO" ":" "qnxnto[";
00426 #endif
00427
00428 #if defined(__CRAYXT_COMPUTE_LINUX_TARGET)
00429 char const *info_cray = "INFO" ":" "compiler_wrapper[CrayPrgEnv]";
00430 #endif
00431
00432 #define STRINGIFY_HELPER(X) #X
00433 #define STRINGIFY(X) STRINGIFY_HELPER(X)
00434
00435 /* Identify known platforms by name. */
00436 #if defined(__linux) || defined(__linux__) || defined(linux)
00437 # define PLATFORM_ID "Linux"
00438
00439 #elif defined(__MSYS__)
00440 # define PLATFORM_ID "MSYS"
00441
00442 #elif defined(__CYGWIN__)
00443 # define PLATFORM_ID "Cygwin"
00444
00445 #elif defined(__MINGW32__)
00446 # define PLATFORM_ID "MinGW"
00447
00448 #elif defined(__APPLE__)
00449 # define PLATFORM_ID "Darwin"
00450
00451 #elif defined(__WIN32) || defined(__WIN32__) || defined(WIN32)
00452 # define PLATFORM_ID "Windows"
00453
00454 #elif defined(__FreeBSD__) || defined(__FreeBSD)
00455 # define PLATFORM_ID "FreeBSD"
00456
00457 #elif defined(__NetBSD__) || defined(__NetBSD)
00458 # define PLATFORM_ID "NetBSD"
00459
00460 #elif defined(__OpenBSD__) || defined(__OPENBSD)
00461 # define PLATFORM_ID "OpenBSD"
00462
00463 #elif defined(__sun) || defined(sun)
00464 # define PLATFORM_ID "SunOS"
00465
00466 #elif defined(_AIX) || defined(__AIX) || defined(__AIX__) || defined(__aix) || defined(__aix__)
00467 # define PLATFORM_ID "AIX"
00468
00469 #elif defined(__hpux) || defined(__hpux__)
00470 # define PLATFORM_ID "HP-UX"
00471
00472 #elif defined(__HAIKU__)
00473 # define PLATFORM_ID "Haiku"
00474
00475 #elif defined(__BeOS) || defined(__BEOS__) || defined(_BEOS)
00476 # define PLATFORM_ID "BeOS"
00477
00478 #elif defined(__QNX__) || defined(__QNXNTO__)
00479 # define PLATFORM_ID "QNX"
00480
00481 #elif defined(__tru64) || defined(_tru64) || defined(__TRU64__)
00482 # define PLATFORM_ID "Tru64"
00483
00484 #elif defined(__riscos) || defined(__riscos__)
00485 # define PLATFORM_ID "RISCos"
00486
00487 #elif defined(__sinix) || defined(__sinix__) || defined(__SINIX__)
00488 # define PLATFORM_ID "SINIX"
00489
00490 #elif defined(__UNIX_SV__)
00491 # define PLATFORM_ID "UNIX_SV"
00492
00493 #elif defined(__bsdos__)
00494 # define PLATFORM_ID "BSDOS"
00495
00496 #elif defined(_MPRAS) || defined(MPRAS)
00497 # define PLATFORM_ID "MP-RAS"
00498
00499 #elif defined(__osf) || defined(__osf__)
00500 # define PLATFORM_ID "OSF1"
00501

```



```

00502 #elif defined(_SCO_SV) || defined(SCO_SV) || defined(sco_sv)
00503 # define PLATFORM_ID "SCO_SV"
00504
00505 #elif defined(__ultrix) || defined(__ultrix__) || defined(_ULTRIX)
00506 # define PLATFORM_ID "ULTRIX"
00507
00508 #elif defined(__XENIX__) || defined(_XENIX) || defined(XENIX)
00509 # define PLATFORM_ID "Xenix"
00510
00511 #elif defined(__WATCOMC__)
00512 # if defined(__LINUX__)
00513 #   define PLATFORM_ID "Linux"
00514
00515 # elif defined(__DOS__)
00516 #   define PLATFORM_ID "DOS"
00517
00518 # elif defined(__OS2__)
00519 #   define PLATFORM_ID "OS2"
00520
00521 # elif defined(__WINDOWS__)
00522 #   define PLATFORM_ID "Windows3x"
00523
00524 # elif defined(__VXWORKS__)
00525 #   define PLATFORM_ID "VxWorks"
00526
00527 # else /* unknown platform */
00528 #   define PLATFORM_ID
00529 # endif
00530
00531 #elif defined(__INTEGRITY)
00532 # if defined(INT_178B)
00533 #   define PLATFORM_ID "Integrity178"
00534
00535 # else /* regular Integrity */
00536 #   define PLATFORM_ID "Integrity"
00537 # endif
00538
00539 #else /* unknown platform */
00540 # define PLATFORM_ID
00541
00542 #endif
00543
00544 /* For windows compilers MSVC and Intel we can determine
00545    the architecture of the compiler being used. This is because
00546    the compilers do not have flags that can change the architecture,
00547    but rather depend on which compiler is being used
00548 */
00549 #if defined(_WIN32) && defined(_MSC_VER)
00550 # if defined(_M_IA64)
00551 #   define ARCHITECTURE_ID "IA64"
00552
00553 # elif defined(_M_ARM64EC)
00554 #   define ARCHITECTURE_ID "ARM64EC"
00555
00556 # elif defined(_M_X64) || defined(_M_AMD64)
00557 #   define ARCHITECTURE_ID "x64"
00558
00559 # elif defined(_M_IX86)
00560 #   define ARCHITECTURE_ID "X86"
00561
00562 # elif defined(_M_ARM64)
00563 #   define ARCHITECTURE_ID "ARM64"
00564
00565 # elif defined(_M_ARM)
00566 #   if _M_ARM == 4
00567 #     define ARCHITECTURE_ID "ARMV4I"
00568 #   elif _M_ARM == 5
00569 #     define ARCHITECTURE_ID "ARMV5I"
00570 #   else
00571 #     define ARCHITECTURE_ID "ARMV" STRINGIFY(_M_ARM)
00572 #   endif
00573
00574 # elif defined(_M_MIPS)
00575 #   define ARCHITECTURE_ID "MIPS"
00576
00577 # elif defined(_M_SH)
00578 #   define ARCHITECTURE_ID "SHx"
00579
00580 # else /* unknown architecture */
00581 #   define ARCHITECTURE_ID ""
00582 # endif
00583
00584 #elif defined(__WATCOMC__)
00585 # if defined(_M_I86)
00586 #   define ARCHITECTURE_ID "I86"
00587
00588 # elif defined(_M_IX86)

```

```

00589 # define ARCHITECTURE_ID "X86"
00590
00591 # else /* unknown architecture */
00592 # define ARCHITECTURE_ID ""
00593 # endif
00594
00595 #elif defined(__IAR_SYSTEMS_ICC__) || defined(__IAR_SYSTEMS_ICC)
00596 # if defined(__ICCARM__)
00597 # define ARCHITECTURE_ID "ARM"
00598
00599 # elif defined(__ICCRX__)
00600 # define ARCHITECTURE_ID "RX"
00601
00602 # elif defined(__ICCRH850__)
00603 # define ARCHITECTURE_ID "RH850"
00604
00605 # elif defined(__ICCRL78__)
00606 # define ARCHITECTURE_ID "RL78"
00607
00608 # elif defined(__ICCRISCV__)
00609 # define ARCHITECTURE_ID "RISCV"
00610
00611 # elif defined(__ICCAVR__)
00612 # define ARCHITECTURE_ID "AVR"
00613
00614 # elif defined(__ICC430__)
00615 # define ARCHITECTURE_ID "MSP430"
00616
00617 # elif defined(__ICCV850__)
00618 # define ARCHITECTURE_ID "V850"
00619
00620 # elif defined(__ICC8051__)
00621 # define ARCHITECTURE_ID "8051"
00622
00623 # elif defined(__IC CSTM8__)
00624 # define ARCHITECTURE_ID "STM8"
00625
00626 # else /* unknown architecture */
00627 # define ARCHITECTURE_ID ""
00628 # endif
00629
00630 #elif defined(__ghs__)
00631 # if defined(__PPC64__)
00632 # define ARCHITECTURE_ID "PPC64"
00633
00634 # elif defined(__ppc__)
00635 # define ARCHITECTURE_ID "PPC"
00636
00637 # elif defined(__ARM__)
00638 # define ARCHITECTURE_ID "ARM"
00639
00640 # elif defined(__x86_64__)
00641 # define ARCHITECTURE_ID "x64"
00642
00643 # elif defined(__i386__)
00644 # define ARCHITECTURE_ID "X86"
00645
00646 # else /* unknown architecture */
00647 # define ARCHITECTURE_ID ""
00648 # endif
00649
00650 #elif defined(__TI_COMPILER_VERSION__)
00651 # if defined(__TI_ARM__)
00652 # define ARCHITECTURE_ID "ARM"
00653
00654 # elif defined(__MSP430__)
00655 # define ARCHITECTURE_ID "MSP430"
00656
00657 # elif defined(__TMS320C28XX__)
00658 # define ARCHITECTURE_ID "TMS320C28x"
00659
00660 # elif defined(__TMS320C6X__) || defined(_TMS320C6X)
00661 # define ARCHITECTURE_ID "TMS320C6x"
00662
00663 # else /* unknown architecture */
00664 # define ARCHITECTURE_ID ""
00665 # endif
00666
00667 #else
00668 # define ARCHITECTURE_ID
00669 #endif
00670
00671 /* Convert integer to decimal digit literals. */
00672 #define DEC(n) \
00673 ('0' + ((n) / 10000000)%10), \
00674 ('0' + ((n) / 1000000)%10), \
00675 ('0' + ((n) / 100000)%10), \

```

```

00676 ('0' + ((n) / 10000)%10)), \
00677 ('0' + ((n) / 1000)%10)), \
00678 ('0' + ((n) / 100)%10)), \
00679 ('0' + ((n) / 10)%10)), \
00680 ('0' + (n) % 10))
00681
00682 /* Convert integer to hex digit literals. */
00683 #define HEX(n) \
00684 ('0' + ((n)>>28 & 0xF)), \
00685 ('0' + ((n)>>24 & 0xF)), \
00686 ('0' + ((n)>>20 & 0xF)), \
00687 ('0' + ((n)>>16 & 0xF)), \
00688 ('0' + ((n)>>12 & 0xF)), \
00689 ('0' + ((n)>>8 & 0xF)), \
00690 ('0' + ((n)>>4 & 0xF)), \
00691 ('0' + ((n) & 0xF))
00692
00693 /* Construct a string literal encoding the version number. */
00694 #ifdef COMPILER_VERSION
00695 char const* info_version = "INFO" ":" "compiler_version[" COMPILER_VERSION "];"
00696
00697 /* Construct a string literal encoding the version number components. */
00698 #elif defined(COMPILER_VERSION_MAJOR)
00699 char const info_version[] = {
00700 'I','N','F','O',':',
00701 'c','o','m','p','i','l','e','r','_','v','e','r','s','i','o','n',' ',
00702 COMPILER_VERSION_MAJOR,
00703 # ifdef COMPILER_VERSION_MINOR
00704 '.', COMPILER_VERSION_MINOR,
00705 # ifdef COMPILER_VERSION_PATCH
00706 '.', COMPILER_VERSION_PATCH,
00707 # ifdef COMPILER_VERSION_TWEAK
00708 '.', COMPILER_VERSION_TWEAK,
00709 # endif
00710 # endif
00711 # endif
00712 '}', '\0';
00713 #endif
00714
00715 /* Construct a string literal encoding the internal version number. */
00716 #ifdef COMPILER_VERSION_INTERNAL
00717 char const info_version_internal[] = {
00718 'I','N','F','O',':',
00719 'c','o','m','p','i','l','e','r','_','v','e','r','s','i','o','n','_','i',
00720 'n','t','e','r','n','a','l',' ',
00721 COMPILER_VERSION_INTERNAL, '}', '\0';
00722 #elif defined(COMPILER_VERSION_INTERNAL_STR)
00723 char const* info_version_internal = "INFO" ":" "compiler_version_internal["
COMPILER_VERSION_INTERNAL_STR "];"
00724 #endif
00725
00726 /* Construct a string literal encoding the version number components. */
00727 #ifdef SIMULATE_VERSION_MAJOR
00728 char const info_simulate_version[] = {
00729 'I','N','F','O',':',
00730 's','i','m','u','l','a','t','e','r','_','v','e','r','s','i','o','n',' ',
00731 SIMULATE_VERSION_MAJOR,
00732 # ifdef SIMULATE_VERSION_MINOR
00733 '.', SIMULATE_VERSION_MINOR,
00734 # ifdef SIMULATE_VERSION_PATCH
00735 '.', SIMULATE_VERSION_PATCH,
00736 # ifdef SIMULATE_VERSION_TWEAK
00737 '.', SIMULATE_VERSION_TWEAK,
00738 # endif
00739 # endif
00740 # endif
00741 '}', '\0';
00742 #endif
00743
00744 /* Construct the string literal in pieces to prevent the source from
00745 getting matched. Store it in a pointer rather than an array
00746 because some compilers will just produce instructions to fill the
00747 array rather than assigning a pointer to a static array. */
00748 char const* info_platform = "INFO" ":" "platform[" PLATFORM_ID "];"
00749 char const* info_arch = "INFO" ":" "arch[" ARCHITECTURE_ID "];"
00750
00751
00752
00753 #if !defined(__STDC__) && !defined(__clang__)
00754 # if defined(_MSC_VER) || defined(_ibmxl_) || defined(__IBMC__)
00755 # define C_DIALECT "90"
00756 # else
00757 # define C_DIALECT
00758 # endif
00759 #elif __STDC_VERSION__ > 201710L
00760 # define C_DIALECT "23"
00761 #elif __STDC_VERSION__ >= 201710L

```

```

00762 # define C_DIALECT "17"
00763 #elif __STDC_VERSION__ >= 201000L
00764 # define C_DIALECT "11"
00765 #elif __STDC_VERSION__ >= 199901L
00766 # define C_DIALECT "99"
00767 #else
00768 # define C_DIALECT "90"
00769 #endif
00770 const char* info_language_dialect_default =
00771     "INFO" ":" "dialect_default[" C_DIALECT " ]";
00772
00773 /*-----*/
00774
00775 #ifdef ID_VOID_MAIN
00776 void main() {}
00777 #else
00778 # if defined(__CLASSIC_C__)
00779 int main(argc, argv) int argc; char *argv[];
00780 # else
00781 int main(int argc, char* argv[])
00782 # endif
00783 {
00784     int require = 0;
00785     require += info_compiler[argc];
00786     require += info_platform[argc];
00787     require += info_arch[argc];
00788 #ifdef COMPILER_VERSION_MAJOR
00789     require += info_version[argc];
00790 #endif
00791 #ifdef COMPILER_VERSION_INTERNAL
00792     require += info_version_internal[argc];
00793 #endif
00794 #ifdef SIMULATE_ID
00795     require += info_simulate[argc];
00796 #endif
00797 #ifdef SIMULATE_VERSION_MAJOR
00798     require += info_simulate_version[argc];
00799 #endif
00800 #if defined(__CRAYXT_COMPUTE_LINUX_TARGET)
00801     require += info_cray[argc];
00802 #endif
00803     require += info_language_dialect_default[argc];
00804     (void)argv;
00805     return require;
00806 }
00807 #endif

```

6.3 D:/Documents/Final/Mandelbrot_Cpp_Final/cmake-build-debug/↵ CMakeFiles/3.21.1/CompilerIdCXX/CMakeCXXCompilerId.cpp File Reference

Macros

- #define `__has_include(x)` 0
- #define `COMPILER_ID` ""
- #define `STRINGIFY_HELPER(X)` #X
- #define `STRINGIFY(X)` `STRINGIFY_HELPER(X)`
- #define `PLATFORM_ID`
- #define `ARCHITECTURE_ID`
- #define `DEC(n)`
- #define `HEX(n)`
- #define `CXX_STD` `__cplusplus`

Functions

- int `main` (int argc, char *argv[])

Variables

- char const * [info_compiler](#) = "INFO" ":" "compiler[" COMPILER_ID "]"
- char const * [info_platform](#) = "INFO" ":" "platform[" PLATFORM_ID "]"
- char const * [info_arch](#) = "INFO" ":" "arch[" ARCHITECTURE_ID "]"
- const char * [info_language_dialect_default](#)

6.3.1 Macro Definition Documentation

6.3.1.1 __has_include

```
#define __has_include(  
    x ) 0
```

Definition at line 11 of file [CMakeCXXCompilerId.cpp](#).

6.3.1.2 ARCHITECTURE_ID

```
#define ARCHITECTURE_ID
```

Definition at line 653 of file [CMakeCXXCompilerId.cpp](#).

6.3.1.3 COMPILER_ID

```
#define COMPILER_ID ""
```

Definition at line 397 of file [CMakeCXXCompilerId.cpp](#).

6.3.1.4 CXX_STD

```
#define CXX_STD __cplusplus
```

Definition at line 751 of file [CMakeCXXCompilerId.cpp](#).

6.3.1.5 DEC

```
#define DEC(  
    n )
```

Value:

```
('0' + ((n) / 10000000) % 10), \  
( '0' + ((n) / 1000000) % 10), \  
( '0' + ((n) / 100000) % 10), \  
( '0' + ((n) / 10000) % 10), \  
( '0' + ((n) / 1000) % 10), \  
( '0' + ((n) / 100) % 10), \  
( '0' + ((n) / 10) % 10), \  
( '0' + ((n) % 10))
```

Definition at line 657 of file [CMakeCXXCompilerId.cpp](#).

6.3.1.6 HEX

```
#define HEX(  
    n )
```

Value:

```
('0' + ((n) >> 28 & 0xF)), \  
( '0' + ((n) >> 24 & 0xF)), \  
( '0' + ((n) >> 20 & 0xF)), \  
( '0' + ((n) >> 16 & 0xF)), \  
( '0' + ((n) >> 12 & 0xF)), \  
( '0' + ((n) >> 8 & 0xF)), \  
( '0' + ((n) >> 4 & 0xF)), \  
( '0' + ((n) & 0xF))
```

Definition at line 668 of file [CMakeCXXCompilerId.cpp](#).

6.3.1.7 PLATFORM_ID

```
#define PLATFORM_ID
```

Definition at line 525 of file [CMakeCXXCompilerId.cpp](#).

6.3.1.8 STRINGIFY

```
#define STRINGIFY(  
    X ) STRINGIFY_HELPER(X)
```

Definition at line 418 of file [CMakeCXXCompilerId.cpp](#).

6.3.1.9 STRINGIFY_HELPER

```
#define STRINGIFY_HELPER(  
    X ) #X
```

Definition at line 417 of file [CMakeCXXCompilerId.cpp](#).

6.3.2 Function Documentation

6.3.2.1 main()

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

Definition at line 772 of file [CMakeCXXCompilerId.cpp](#).

```
00773 {  
00774     int require = 0;  
00775     require += info_compiler[argc];  
00776     require += info_platform[argc];  
00777     #ifdef COMPILER_VERSION_MAJOR  
00778     require += info_version[argc];  
00779     #endif  
00780     #ifdef COMPILER_VERSION_INTERNAL  
00781     require += info_version_internal[argc];  
00782     #endif  
00783     #ifdef SIMULATE_ID  
00784     require += info_simulate[argc];  
00785     #endif  
00786     #ifdef SIMULATE_VERSION_MAJOR  
00787     require += info_simulate_version[argc];  
00788     #endif  
00789     #if defined(__CRAYXT_COMPUTE_LINUX_TARGET)  
00790     require += info_cray[argc];  
00791     #endif  
00792     require += info_language_dialect_default[argc];  
00793     (void)argv;  
00794     return require;  
00795 }
```

6.3.3 Variable Documentation

6.3.3.1 info_arch

```
char const* info_arch = "INFO" ":" "arch[" ARCHITECTURE_ID "]"
```

Definition at line 734 of file [CMakeCXXCompilerId.cpp](#).

6.3.3.2 info_compiler

```
char const* info_compiler = "INFO" ":" "compiler[" COMPILER_ID "]"
```

Definition at line 404 of file [CMakeCXXCompilerId.cpp](#).

6.3.3.3 info_language_dialect_default

```
const char* info_language_dialect_default
```

Initial value:

```
= "INFO" ":" "dialect_default["  
    "98"  
"]"
```

Definition at line 754 of file [CMakeCXXCompilerId.cpp](#).

6.3.3.4 info_platform

```
char const* info_platform = "INFO" ":" "platform[" PLATFORM_ID "]"
```

Definition at line 733 of file [CMakeCXXCompilerId.cpp](#).

6.4 CMakeCXXCompilerId.cpp

[Go to the documentation of this file.](#)

```
00001 /* This source file must have a .cpp extension so that all C++ compilers
00002    recognize the extension without flags. Borland does not know .cxx for
00003    example. */
00004 #ifndef __cplusplus
00005 # error "A C compiler has been selected for C++."
00006 #endif
00007
00008 #if !defined(__has_include)
00009 /* If the compiler does not have __has_include, pretend the answer is
00010    always no. */
00011 # define __has_include(x) 0
00012 #endif
00013
00014
00015 /* Version number components: V=Version, R=Revision, P=Patch
00016    Version date components: YYYY=Year, MM=Month, DD=Day */
00017
00018 #if defined(__COMO__)
00019 # define COMPILER_ID "Comeau"
00020 /* __COMO_VERSION__ = VRR */
00021 # define COMPILER_VERSION_MAJOR DEC(__COMO_VERSION__ / 100)
00022 # define COMPILER_VERSION_MINOR DEC(__COMO_VERSION__ % 100)
00023
00024 #elif defined(__INTEL_COMPILER) || defined(__ICC)
00025 # define COMPILER_ID "Intel"
00026 # if defined(_MSC_VER)
00027 #   define SIMULATE_ID "MSVC"
00028 # endif
00029 # if defined(__GNUC__)
00030 #   define SIMULATE_ID "GNU"
00031 # endif
00032 /* __INTEL_COMPILER = VRP prior to 2021, and then VVVV for 2021 and later,
00033    except that a few beta releases use the old format with V=2021. */
00034 # if __INTEL_COMPILER < 2021 || __INTEL_COMPILER == 202110 || __INTEL_COMPILER == 202111
```



```

00035 # define COMPILER_VERSION_MAJOR DEC(__INTEL_COMPILER/100)
00036 # define COMPILER_VERSION_MINOR DEC(__INTEL_COMPILER/10 % 10)
00037 # if defined(__INTEL_COMPILER_UPDATE)
00038 #   define COMPILER_VERSION_PATCH DEC(__INTEL_COMPILER_UPDATE)
00039 # else
00040 #   define COMPILER_VERSION_PATCH DEC(__INTEL_COMPILER % 10)
00041 # endif
00042 # else
00043 #   define COMPILER_VERSION_MAJOR DEC(__INTEL_COMPILER)
00044 #   define COMPILER_VERSION_MINOR DEC(__INTEL_COMPILER_UPDATE)
00045 #   /* The third version component from --version is an update index,
00046 #      but no macro is provided for it. */
00047 #   define COMPILER_VERSION_PATCH DEC(0)
00048 # endif
00049 # if defined(__INTEL_COMPILER_BUILD_DATE)
00050 #   /* __INTEL_COMPILER_BUILD_DATE = YYYYMMDD */
00051 #   define COMPILER_VERSION_TWEAK DEC(__INTEL_COMPILER_BUILD_DATE)
00052 # endif
00053 # if defined(_MSC_VER)
00054 #   /* _MSC_VER = VVRR */
00055 #   define SIMULATE_VERSION_MAJOR DEC(_MSC_VER / 100)
00056 #   define SIMULATE_VERSION_MINOR DEC(_MSC_VER % 100)
00057 # endif
00058 # if defined(__GNUC__)
00059 #   define SIMULATE_VERSION_MAJOR DEC(__GNUC__)
00060 # elif defined(__GNUG__)
00061 #   define SIMULATE_VERSION_MAJOR DEC(__GNUG__)
00062 # endif
00063 # if defined(__GNUC_MINOR__)
00064 #   define SIMULATE_VERSION_MINOR DEC(__GNUC_MINOR__)
00065 # endif
00066 # if defined(__GNUC_PATCHLEVEL__)
00067 #   define SIMULATE_VERSION_PATCH DEC(__GNUC_PATCHLEVEL__)
00068 # endif
00069
00070 #elif (defined(__clang__) && defined(__INTEL_CLANG_COMPILER)) || defined(__INTEL_LLVM_COMPILER)
00071 # define COMPILER_ID "IntelLLVM"
00072 #if defined(_MSC_VER)
00073 # define SIMULATE_ID "MSVC"
00074 #endif
00075 #if defined(__GNUC__)
00076 # define SIMULATE_ID "GNU"
00077 #endif
00078 /* __INTEL_LLVM_COMPILER = VVVVRRP prior to 2021.2.0, VVVVRRPP for 2021.2.0 and
00079  * later. Look for 6 digit vs. 8 digit version number to decide encoding.
00080  * VVVV is no smaller than the current year when a version is released.
00081  */
00082 #if __INTEL_LLVM_COMPILER < 1000000L
00083 # define COMPILER_VERSION_MAJOR DEC(__INTEL_LLVM_COMPILER/100)
00084 # define COMPILER_VERSION_MINOR DEC(__INTEL_LLVM_COMPILER/10 % 10)
00085 # define COMPILER_VERSION_PATCH DEC(__INTEL_LLVM_COMPILER % 10)
00086 #else
00087 # define COMPILER_VERSION_MAJOR DEC(__INTEL_LLVM_COMPILER/10000)
00088 # define COMPILER_VERSION_MINOR DEC(__INTEL_LLVM_COMPILER/100 % 100)
00089 # define COMPILER_VERSION_PATCH DEC(__INTEL_LLVM_COMPILER % 100)
00090 #endif
00091 #if defined(_MSC_VER)
00092 #   /* _MSC_VER = VVRR */
00093 #   define SIMULATE_VERSION_MAJOR DEC(_MSC_VER / 100)
00094 #   define SIMULATE_VERSION_MINOR DEC(_MSC_VER % 100)
00095 #endif
00096 #if defined(__GNUC__)
00097 #   define SIMULATE_VERSION_MAJOR DEC(__GNUC__)
00098 #elif defined(__GNUG__)
00099 #   define SIMULATE_VERSION_MAJOR DEC(__GNUG__)
00100 #endif
00101 #if defined(__GNUC_MINOR__)
00102 #   define SIMULATE_VERSION_MINOR DEC(__GNUC_MINOR__)
00103 #endif
00104 #if defined(__GNUC_PATCHLEVEL__)
00105 #   define SIMULATE_VERSION_PATCH DEC(__GNUC_PATCHLEVEL__)
00106 #endif
00107
00108 #elif defined(__PATHCC__)
00109 # define COMPILER_ID "PathScale"
00110 # define COMPILER_VERSION_MAJOR DEC(__PATHCC__)
00111 # define COMPILER_VERSION_MINOR DEC(__PATHCC_MINOR__)
00112 # if defined(__PATHCC_PATCHLEVEL__)
00113 #   define COMPILER_VERSION_PATCH DEC(__PATHCC_PATCHLEVEL__)
00114 # endif
00115
00116 #elif defined(__BORLANDC__) && defined(__CODEGEARC_VERSION__)
00117 # define COMPILER_ID "Embarcadero"
00118 # define COMPILER_VERSION_MAJOR HEX(__CODEGEARC_VERSION__>24 & 0x00FF)
00119 # define COMPILER_VERSION_MINOR HEX(__CODEGEARC_VERSION__>16 & 0x00FF)
00120 # define COMPILER_VERSION_PATCH DEC(__CODEGEARC_VERSION__ & 0xFFFF)
00121

```

```

00122 #elif defined(__BORLANDC__)
00123 # define COMPILER_ID "Borland"
00124 /* __BORLANDC__ = 0xVRR */
00125 # define COMPILER_VERSION_MAJOR HEX(__BORLANDC__>8)
00126 # define COMPILER_VERSION_MINOR HEX(__BORLANDC__ & 0xFF)
00127
00128 #elif defined(__WATCOMC__) && __WATCOMC__ < 1200
00129 # define COMPILER_ID "Watcom"
00130 /* __WATCOMC__ = VVRR */
00131 # define COMPILER_VERSION_MAJOR DEC(__WATCOMC__ / 100)
00132 # define COMPILER_VERSION_MINOR DEC((__WATCOMC__ / 10) % 10)
00133 # if (__WATCOMC__ % 10) > 0
00134 #   define COMPILER_VERSION_PATCH DEC(__WATCOMC__ % 10)
00135 # endif
00136
00137 #elif defined(__WATCOMC__)
00138 # define COMPILER_ID "OpenWatcom"
00139 /* __WATCOMC__ = VVRR + 1100 */
00140 # define COMPILER_VERSION_MAJOR DEC((__WATCOMC__ - 1100) / 100)
00141 # define COMPILER_VERSION_MINOR DEC((__WATCOMC__ / 10) % 10)
00142 # if (__WATCOMC__ % 10) > 0
00143 #   define COMPILER_VERSION_PATCH DEC(__WATCOMC__ % 10)
00144 # endif
00145
00146 #elif defined(__SUNPRO_CC)
00147 # define COMPILER_ID "SunPro"
00148 # if __SUNPRO_CC >= 0x5100
00149 /* __SUNPRO_CC = 0xVRRP */
00150 #   define COMPILER_VERSION_MAJOR HEX(__SUNPRO_CC>12)
00151 #   define COMPILER_VERSION_MINOR HEX(__SUNPRO_CC>4 & 0xFF)
00152 #   define COMPILER_VERSION_PATCH HEX(__SUNPRO_CC & 0xF)
00153 # else
00154 /* __SUNPRO_CC = 0xVRP */
00155 #   define COMPILER_VERSION_MAJOR HEX(__SUNPRO_CC>8)
00156 #   define COMPILER_VERSION_MINOR HEX(__SUNPRO_CC>4 & 0xF)
00157 #   define COMPILER_VERSION_PATCH HEX(__SUNPRO_CC & 0xF)
00158 # endif
00159
00160 #elif defined(__HP_aCC)
00161 # define COMPILER_ID "HP"
00162 /* __HP_aCC = VVRRPP */
00163 # define COMPILER_VERSION_MAJOR DEC(__HP_aCC/10000)
00164 # define COMPILER_VERSION_MINOR DEC(__HP_aCC/100 % 100)
00165 # define COMPILER_VERSION_PATCH DEC(__HP_aCC % 100)
00166
00167 #elif defined(__DECCXX)
00168 # define COMPILER_ID "Compaq"
00169 /* __DECCXX_VER = VVRRTPPPP */
00170 # define COMPILER_VERSION_MAJOR DEC(__DECCXX_VER/10000000)
00171 # define COMPILER_VERSION_MINOR DEC(__DECCXX_VER/100000 % 100)
00172 # define COMPILER_VERSION_PATCH DEC(__DECCXX_VER % 10000)
00173
00174 #elif defined(__IBMCPP__) && defined(__COMPILER_VER__)
00175 # define COMPILER_ID "zOS"
00176 /* __IBMCPP__ = VRP */
00177 # define COMPILER_VERSION_MAJOR DEC(__IBMCPP__/100)
00178 # define COMPILER_VERSION_MINOR DEC(__IBMCPP__/10 % 10)
00179 # define COMPILER_VERSION_PATCH DEC(__IBMCPP__ % 10)
00180
00181 #elif defined(__ibmxl__) && defined(__clang__)
00182 # define COMPILER_ID "XLClang"
00183 # define COMPILER_VERSION_MAJOR DEC(__ibmxl_version__)
00184 # define COMPILER_VERSION_MINOR DEC(__ibmxl_release__)
00185 # define COMPILER_VERSION_PATCH DEC(__ibmxl_modification__)
00186 # define COMPILER_VERSION_TWEAK DEC(__ibmxl_ptf_fix_level__)
00187
00188
00189 #elif defined(__IBMCPP__) && !defined(__COMPILER_VER__) && __IBMCPP__ >= 800
00190 # define COMPILER_ID "XL"
00191 /* __IBMCPP__ = VRP */
00192 # define COMPILER_VERSION_MAJOR DEC(__IBMCPP__/100)
00193 # define COMPILER_VERSION_MINOR DEC(__IBMCPP__/10 % 10)
00194 # define COMPILER_VERSION_PATCH DEC(__IBMCPP__ % 10)
00195
00196 #elif defined(__IBMCPP__) && !defined(__COMPILER_VER__) && __IBMCPP__ < 800
00197 # define COMPILER_ID "VisualAge"
00198 /* __IBMCPP__ = VRP */
00199 # define COMPILER_VERSION_MAJOR DEC(__IBMCPP__/100)
00200 # define COMPILER_VERSION_MINOR DEC(__IBMCPP__/10 % 10)
00201 # define COMPILER_VERSION_PATCH DEC(__IBMCPP__ % 10)
00202
00203 #elif defined(__NVCOMPILER)
00204 # define COMPILER_ID "NVHPC"
00205 # define COMPILER_VERSION_MAJOR DEC(__NVCOMPILER_MAJOR__)
00206 # define COMPILER_VERSION_MINOR DEC(__NVCOMPILER_MINOR__)
00207 # if defined(__NVCOMPILER_PATCHLEVEL__)
00208 #   define COMPILER_VERSION_PATCH DEC(__NVCOMPILER_PATCHLEVEL__)

```

```

00209 # endif
00210
00211 #elif defined(__PGI)
00212 # define COMPILER_ID "PGI"
00213 # define COMPILER_VERSION_MAJOR DEC(__PGIC__)
00214 # define COMPILER_VERSION_MINOR DEC(__PGIC_MINOR__)
00215 # if defined(__PGIC_PATCHLEVEL__)
00216 #   define COMPILER_VERSION_PATCH DEC(__PGIC_PATCHLEVEL__)
00217 # endif
00218
00219 #elif defined(__CRAYC)
00220 # define COMPILER_ID "Cray"
00221 # define COMPILER_VERSION_MAJOR DEC(_RELEASE_MAJOR)
00222 # define COMPILER_VERSION_MINOR DEC(_RELEASE_MINOR)
00223
00224 #elif defined(__TI_COMPILER_VERSION__)
00225 # define COMPILER_ID "TI"
00226 /* __TI_COMPILER_VERSION__ = VVRRRRPPP */
00227 # define COMPILER_VERSION_MAJOR DEC(__TI_COMPILER_VERSION__/1000000)
00228 # define COMPILER_VERSION_MINOR DEC(__TI_COMPILER_VERSION__/1000 % 1000)
00229 # define COMPILER_VERSION_PATCH DEC(__TI_COMPILER_VERSION__ % 1000)
00230
00231 #elif defined(__CLANG_FUJITSU)
00232 # define COMPILER_ID "FujitsuClang"
00233 # define COMPILER_VERSION_MAJOR DEC(__FCC_major__)
00234 # define COMPILER_VERSION_MINOR DEC(__FCC_minor__)
00235 # define COMPILER_VERSION_PATCH DEC(__FCC_patchlevel__)
00236 # define COMPILER_VERSION_INTERNAL_STR __clang_version__
00237
00238
00239 #elif defined(__FUJITSU)
00240 # define COMPILER_ID "Fujitsu"
00241 # if defined(__FCC_version__)
00242 #   define COMPILER_VERSION __FCC_version__
00243 # elif defined(__FCC_major__)
00244 #   define COMPILER_VERSION_MAJOR DEC(__FCC_major__)
00245 #   define COMPILER_VERSION_MINOR DEC(__FCC_minor__)
00246 #   define COMPILER_VERSION_PATCH DEC(__FCC_patchlevel__)
00247 # endif
00248 # if defined(__fcc_version)
00249 #   define COMPILER_VERSION_INTERNAL DEC(__fcc_version)
00250 # elif defined(__FCC_VERSION)
00251 #   define COMPILER_VERSION_INTERNAL DEC(__FCC_VERSION)
00252 # endif
00253
00254
00255 #elif defined(__ghs__)
00256 # define COMPILER_ID "GHS"
00257 /* __GHS_VERSION_NUMBER = VVVVRP */
00258 # ifdef __GHS_VERSION_NUMBER
00259 #   define COMPILER_VERSION_MAJOR DEC(__GHS_VERSION_NUMBER / 100)
00260 #   define COMPILER_VERSION_MINOR DEC(__GHS_VERSION_NUMBER / 10 % 10)
00261 #   define COMPILER_VERSION_PATCH DEC(__GHS_VERSION_NUMBER % 10)
00262 # endif
00263
00264 #elif defined(__SCO_VERSION__)
00265 # define COMPILER_ID "SCO"
00266
00267 #elif defined(__ARMCC_VERSION) && !defined(__clang__)
00268 # define COMPILER_ID "ARMCC"
00269 #if __ARMCC_VERSION >= 1000000
00270 /* __ARMCC_VERSION = VRRPPPP */
00271 # define COMPILER_VERSION_MAJOR DEC(__ARMCC_VERSION/1000000)
00272 # define COMPILER_VERSION_MINOR DEC(__ARMCC_VERSION/10000 % 100)
00273 # define COMPILER_VERSION_PATCH DEC(__ARMCC_VERSION % 10000)
00274 #else
00275 /* __ARMCC_VERSION = VRPPPP */
00276 # define COMPILER_VERSION_MAJOR DEC(__ARMCC_VERSION/100000)
00277 # define COMPILER_VERSION_MINOR DEC(__ARMCC_VERSION/10000 % 10)
00278 # define COMPILER_VERSION_PATCH DEC(__ARMCC_VERSION % 10000)
00279 #endif
00280
00281
00282 #elif defined(__clang__) && defined(__apple_build_version__)
00283 # define COMPILER_ID "AppleClang"
00284 # if defined(_MSC_VER)
00285 #   define SIMULATE_ID "MSVC"
00286 # endif
00287 # define COMPILER_VERSION_MAJOR DEC(__clang_major__)
00288 # define COMPILER_VERSION_MINOR DEC(__clang_minor__)
00289 # define COMPILER_VERSION_PATCH DEC(__clang_patchlevel__)
00290 # if defined(_MSC_VER)
00291 /* _MSC_VER = VVRR */
00292 #   define SIMULATE_VERSION_MAJOR DEC(_MSC_VER / 100)
00293 #   define SIMULATE_VERSION_MINOR DEC(_MSC_VER % 100)
00294 # endif
00295 # define COMPILER_VERSION_TWEAK DEC(__apple_build_version__)

```

```

00296
00297 #elif defined(__clang__) && defined(__ARMCOMPILER_VERSION)
00298 # define COMPILER_ID "ARMClang"
00299 # define COMPILER_VERSION_MAJOR DEC(__ARMCOMPILER_VERSION/1000000)
00300 # define COMPILER_VERSION_MINOR DEC(__ARMCOMPILER_VERSION/10000 % 100)
00301 # define COMPILER_VERSION_PATCH DEC(__ARMCOMPILER_VERSION % 10000)
00302 # define COMPILER_VERSION_INTERNAL DEC(__ARMCOMPILER_VERSION)
00303
00304 #elif defined(__clang__) && __has_include(<hip/hip_version.h>)
00305 # define COMPILER_ID "ROCMClang"
00306 # if defined(_MSC_VER)
00307 #   define SIMULATE_ID "MSVC"
00308 # elif defined(__clang__)
00309 #   define SIMULATE_ID "Clang"
00310 # elif defined(__GNUC__)
00311 #   define SIMULATE_ID "GNU"
00312 # endif
00313 # if defined(__clang__) && __has_include(<hip/hip_version.h>)
00314 #   include <hip/hip_version.h>
00315 #   define COMPILER_VERSION_MAJOR DEC(HIP_VERSION_MAJOR)
00316 #   define COMPILER_VERSION_MINOR DEC(HIP_VERSION_MINOR)
00317 #   define COMPILER_VERSION_PATCH DEC(HIP_VERSION_PATCH)
00318 # endif
00319
00320 #elif defined(__clang__)
00321 # define COMPILER_ID "Clang"
00322 # if defined(_MSC_VER)
00323 #   define SIMULATE_ID "MSVC"
00324 # endif
00325 # define COMPILER_VERSION_MAJOR DEC(__clang_major__)
00326 # define COMPILER_VERSION_MINOR DEC(__clang_minor__)
00327 # define COMPILER_VERSION_PATCH DEC(__clang_patchlevel__)
00328 # if defined(_MSC_VER)
00329 /* _MSC_VER = VVRR */
00330 #   define SIMULATE_VERSION_MAJOR DEC(_MSC_VER / 100)
00331 #   define SIMULATE_VERSION_MINOR DEC(_MSC_VER % 100)
00332 # endif
00333
00334 #elif defined(__GNUC__) || defined(__GNUG__)
00335 # define COMPILER_ID "GNU"
00336 # if defined(__GNUC__)
00337 #   define COMPILER_VERSION_MAJOR DEC(__GNUC__)
00338 # else
00339 #   define COMPILER_VERSION_MAJOR DEC(__GNUG__)
00340 # endif
00341 # if defined(__GNUC_MINOR__)
00342 #   define COMPILER_VERSION_MINOR DEC(__GNUC_MINOR__)
00343 # endif
00344 # if defined(__GNUC_PATCHLEVEL__)
00345 #   define COMPILER_VERSION_PATCH DEC(__GNUC_PATCHLEVEL__)
00346 # endif
00347
00348 #elif defined(_MSC_VER)
00349 # define COMPILER_ID "MSVC"
00350 /* _MSC_VER = VVRR */
00351 # define COMPILER_VERSION_MAJOR DEC(_MSC_VER / 100)
00352 # define COMPILER_VERSION_MINOR DEC(_MSC_VER % 100)
00353 # if defined(_MSC_FULL_VER)
00354 #   if _MSC_VER >= 1400
00355 /* _MSC_FULL_VER = VVRRPPPP */
00356 #     define COMPILER_VERSION_PATCH DEC(_MSC_FULL_VER % 100000)
00357 #   else
00358 /* _MSC_FULL_VER = VVRRPPPP */
00359 #     define COMPILER_VERSION_PATCH DEC(_MSC_FULL_VER % 10000)
00360 #   endif
00361 # endif
00362 # if defined(_MSC_BUILD)
00363 #   define COMPILER_VERSION_TWEAK DEC(_MSC_BUILD)
00364 # endif
00365
00366 #elif defined(__VISUALDSPVERSION__) || defined(__ADSPBLACKFIN__) || defined(__ADSPTS__) ||
    defined(__ADSP21000__)
00367 # define COMPILER_ID "ADSP"
00368 #if defined(__VISUALDSPVERSION__)
00369 /* __VISUALDSPVERSION__ = 0xVVRRPP00 */
00370 # define COMPILER_VERSION_MAJOR HEX(__VISUALDSPVERSION__>>24)
00371 # define COMPILER_VERSION_MINOR HEX(__VISUALDSPVERSION__>>16 & 0xFF)
00372 # define COMPILER_VERSION_PATCH HEX(__VISUALDSPVERSION__>>8 & 0xFF)
00373 #endif
00374
00375 #elif defined(__IAR_SYSTEMS_ICC__) || defined(__IAR_SYSTEMS_ICC)
00376 # define COMPILER_ID "IAR"
00377 # if defined(__VER__) && defined(__ICCARM__)
00378 #   define COMPILER_VERSION_MAJOR DEC((__VER__) / 1000000)
00379 #   define COMPILER_VERSION_MINOR DEC(((__VER__) / 1000) % 1000)
00380 #   define COMPILER_VERSION_PATCH DEC((__VER__) % 1000)
00381 #   define COMPILER_VERSION_INTERNAL DEC(__IAR_SYSTEMS_ICC__)

```

```

00382 # elif defined(__VER__) && (defined(__ICCAVR__) || defined(__ICCRX__) || defined(__ICCRH850__) ||
    defined(__ICCRL78__) || defined(__ICC430__) || defined(__ICCRISCV__) || defined(__ICCV850__) ||
    defined(__ICC8051__) || defined(__ICCSSTM8__))
00383 #   define COMPILER_VERSION_MAJOR DEC((__VER__) / 100)
00384 #   define COMPILER_VERSION_MINOR DEC((__VER__) - (((__VER__) / 100)*100))
00385 #   define COMPILER_VERSION_PATCH DEC(__SUBVERSION__)
00386 #   define COMPILER_VERSION_INTERNAL DEC(__IAR_SYSTEMS_ICC__)
00387 # endif
00388
00389 /* These compilers are either not known or too old to define an
00391  identification macro. Try to identify the platform and guess that
00392  it is the native compiler. */
00393 #elif defined(__hpux) || defined(__hpua)
00394 #   define COMPILER_ID "HP"
00395
00396 #else /* unknown compiler */
00397 #   define COMPILER_ID ""
00398 #endif
00399
00400 /* Construct the string literal in pieces to prevent the source from
00401  getting matched. Store it in a pointer rather than an array
00402  because some compilers will just produce instructions to fill the
00403  array rather than assigning a pointer to a static array. */
00404 char const* info_compiler = "INFO" ":" "compiler[" COMPILER_ID "];
00405 #ifdef SIMULATE_ID
00406 char const* info_simulate = "INFO" ":" "simulate[" SIMULATE_ID "];
00407 #endif
00408
00409 #ifdef __QNXNTO__
00410 char const* qnxnto = "INFO" ":" "qnxnto[]";
00411 #endif
00412
00413 #if defined(__CRAYXT_COMPUTE_LINUX_TARGET)
00414 char const* info_cray = "INFO" ":" "compiler_wrapper[CrayPrgEnv]";
00415 #endif
00416
00417 #define STRINGIFY_HELPER(X) #X
00418 #define STRINGIFY(X) STRINGIFY_HELPER(X)
00419
00420 /* Identify known platforms by name. */
00421 #if defined(__linux) || defined(__linux__) || defined(linux)
00422 #   define PLATFORM_ID "Linux"
00423
00424 #elif defined(__MSYS__)
00425 #   define PLATFORM_ID "MSYS"
00426
00427 #elif defined(__CYGWIN__)
00428 #   define PLATFORM_ID "Cygwin"
00429
00430 #elif defined(__MINGW32__)
00431 #   define PLATFORM_ID "MinGW"
00432
00433 #elif defined(__APPLE__)
00434 #   define PLATFORM_ID "Darwin"
00435
00436 #elif defined(__WIN32__) || defined(_WIN32) || defined(WIN32)
00437 #   define PLATFORM_ID "Windows"
00438
00439 #elif defined(__FreeBSD__) || defined(__FreeBSD)
00440 #   define PLATFORM_ID "FreeBSD"
00441
00442 #elif defined(__NetBSD__) || defined(__NetBSD)
00443 #   define PLATFORM_ID "NetBSD"
00444
00445 #elif defined(__OpenBSD__) || defined(__OPENBSD)
00446 #   define PLATFORM_ID "OpenBSD"
00447
00448 #elif defined(__sun) || defined(sun)
00449 #   define PLATFORM_ID "SunOS"
00450
00451 #elif defined(_AIX) || defined(__AIX) || defined(__AIX__) || defined(__aix) || defined(__aix__)
00452 #   define PLATFORM_ID "AIX"
00453
00454 #elif defined(__hpux) || defined(__hpux__)
00455 #   define PLATFORM_ID "HP-UX"
00456
00457 #elif defined(__HAIKU__)
00458 #   define PLATFORM_ID "Haiku"
00459
00460 #elif defined(__BeOS) || defined(__BEOS__) || defined(_BEOS)
00461 #   define PLATFORM_ID "BeOS"
00462
00463 #elif defined(__QNX__) || defined(__QNXNTO__)
00464 #   define PLATFORM_ID "QNX"
00465
00466 #elif defined(__tru64) || defined(_tru64) || defined(__TRU64__)

```

```
00467 # define PLATFORM_ID "Tru64"
00468
00469 #elif defined(__riscos) || defined(__riscos__)
00470 # define PLATFORM_ID "RISCos"
00471
00472 #elif defined(__sinix) || defined(__sinix__) || defined(__SINIX__)
00473 # define PLATFORM_ID "SINIX"
00474
00475 #elif defined(__UNIX_SV__)
00476 # define PLATFORM_ID "UNIX_SV"
00477
00478 #elif defined(__bsdos__)
00479 # define PLATFORM_ID "BSDOS"
00480
00481 #elif defined(__MPRAS) || defined(MPRAS)
00482 # define PLATFORM_ID "MP-RAS"
00483
00484 #elif defined(__osf) || defined(__osf__)
00485 # define PLATFORM_ID "OSF1"
00486
00487 #elif defined(__SCO_SV) || defined(SCO_SV) || defined(sco_sv)
00488 # define PLATFORM_ID "SCO_SV"
00489
00490 #elif defined(__ultrix) || defined(__ultrix__) || defined(ULTRIX)
00491 # define PLATFORM_ID "ULTRIX"
00492
00493 #elif defined(__XENIX__) || defined(_XENIX) || defined(XENIX)
00494 # define PLATFORM_ID "Xenix"
00495
00496 #elif defined(__WATCOMC__)
00497 # if defined(__LINUX__)
00498 #   define PLATFORM_ID "Linux"
00499
00500 # elif defined(__DOS__)
00501 #   define PLATFORM_ID "DOS"
00502
00503 # elif defined(__OS2__)
00504 #   define PLATFORM_ID "OS2"
00505
00506 # elif defined(__WINDOWS__)
00507 #   define PLATFORM_ID "Windows3x"
00508
00509 # elif defined(__VXWORKS__)
00510 #   define PLATFORM_ID "VxWorks"
00511
00512 # else /* unknown platform */
00513 #   define PLATFORM_ID
00514 # endif
00515
00516 #elif defined(__INTEGRITY)
00517 # if defined(INT_178B)
00518 #   define PLATFORM_ID "Integrity178"
00519
00520 # else /* regular Integrity */
00521 #   define PLATFORM_ID "Integrity"
00522 # endif
00523
00524 #else /* unknown platform */
00525 # define PLATFORM_ID
00526
00527 #endif
00528
00529 /* For windows compilers MSVC and Intel we can determine
00530    the architecture of the compiler being used. This is because
00531    the compilers do not have flags that can change the architecture,
00532    but rather depend on which compiler is being used
00533 */
00534 #if defined(_WIN32) && defined(_MSC_VER)
00535 # if defined(_M_IA64)
00536 #   define ARCHITECTURE_ID "IA64"
00537
00538 # elif defined(_M_ARM64EC)
00539 #   define ARCHITECTURE_ID "ARM64EC"
00540
00541 # elif defined(_M_X64) || defined(_M_AMD64)
00542 #   define ARCHITECTURE_ID "x64"
00543
00544 # elif defined(_M_IX86)
00545 #   define ARCHITECTURE_ID "X86"
00546
00547 # elif defined(_M_ARM64)
00548 #   define ARCHITECTURE_ID "ARM64"
00549
00550 # elif defined(_M_ARM)
00551 #   if _M_ARM == 4
00552 #     define ARCHITECTURE_ID "ARMV4I"
00553 #   elif _M_ARM == 5
```

```
00554 #   define ARCHITECTURE_ID "ARMV5I"
00555 #   else
00556 #   define ARCHITECTURE_ID "ARMV" STRINGIFY(_M_ARM)
00557 #   endif
00558
00559 #   elif defined(_M_MIPS)
00560 #   define ARCHITECTURE_ID "MIPS"
00561
00562 #   elif defined(_M_SH)
00563 #   define ARCHITECTURE_ID "SHx"
00564
00565 #   else /* unknown architecture */
00566 #   define ARCHITECTURE_ID ""
00567 #   endif
00568
00569 #elif defined(__WATCOMC__)
00570 #   if defined(_M_I86)
00571 #   define ARCHITECTURE_ID "I86"
00572
00573 #   elif defined(_M_I86)
00574 #   define ARCHITECTURE_ID "X86"
00575
00576 #   else /* unknown architecture */
00577 #   define ARCHITECTURE_ID ""
00578 #   endif
00579
00580 #elif defined(__IAR_SYSTEMS_ICC__) || defined(__IAR_SYSTEMS_ICC)
00581 #   if defined(__ICCARM__)
00582 #   define ARCHITECTURE_ID "ARM"
00583
00584 #   elif defined(__ICCRX__)
00585 #   define ARCHITECTURE_ID "RX"
00586
00587 #   elif defined(__ICCRH850__)
00588 #   define ARCHITECTURE_ID "RH850"
00589
00590 #   elif defined(__ICCRL78__)
00591 #   define ARCHITECTURE_ID "RL78"
00592
00593 #   elif defined(__ICCRISCV__)
00594 #   define ARCHITECTURE_ID "RISCV"
00595
00596 #   elif defined(__ICCAVR__)
00597 #   define ARCHITECTURE_ID "AVR"
00598
00599 #   elif defined(__ICC430__)
00600 #   define ARCHITECTURE_ID "MSP430"
00601
00602 #   elif defined(__ICCV850__)
00603 #   define ARCHITECTURE_ID "V850"
00604
00605 #   elif defined(__ICC8051__)
00606 #   define ARCHITECTURE_ID "8051"
00607
00608 #   elif defined(__IC CSTM8__)
00609 #   define ARCHITECTURE_ID "STM8"
00610
00611 #   else /* unknown architecture */
00612 #   define ARCHITECTURE_ID ""
00613 #   endif
00614
00615 #elif defined(__ghs__)
00616 #   if defined(__PPC64__)
00617 #   define ARCHITECTURE_ID "PPC64"
00618
00619 #   elif defined(__ppc__)
00620 #   define ARCHITECTURE_ID "PPC"
00621
00622 #   elif defined(__ARM__)
00623 #   define ARCHITECTURE_ID "ARM"
00624
00625 #   elif defined(__x86_64__)
00626 #   define ARCHITECTURE_ID "x64"
00627
00628 #   elif defined(__i386__)
00629 #   define ARCHITECTURE_ID "X86"
00630
00631 #   else /* unknown architecture */
00632 #   define ARCHITECTURE_ID ""
00633 #   endif
00634
00635 #elif defined(__TI_COMPILER_VERSION__)
00636 #   if defined(__TI_ARM__)
00637 #   define ARCHITECTURE_ID "ARM"
00638
00639 #   elif defined(__MSP430__)
00640 #   define ARCHITECTURE_ID "MSP430"
```

```

00641
00642 # elif defined(__TMS320C28XX__)
00643 #   define ARCHITECTURE_ID "TMS320C28x"
00644
00645 # elif defined(__TMS320C6X__) || defined(_TMS320C6X)
00646 #   define ARCHITECTURE_ID "TMS320C6x"
00647
00648 # else /* unknown architecture */
00649 #   define ARCHITECTURE_ID ""
00650 # endif
00651
00652 #else
00653 #   define ARCHITECTURE_ID
00654 #endif
00655
00656 /* Convert integer to decimal digit literals. */
00657 #define DEC(n) \
00658   ('0' + ((n) / 10000000) % 10), \
00659   ('0' + ((n) / 1000000) % 10), \
00660   ('0' + ((n) / 100000) % 10), \
00661   ('0' + ((n) / 10000) % 10), \
00662   ('0' + ((n) / 1000) % 10), \
00663   ('0' + ((n) / 100) % 10), \
00664   ('0' + ((n) / 10) % 10), \
00665   ('0' + ((n) % 10))
00666
00667 /* Convert integer to hex digit literals. */
00668 #define HEX(n) \
00669   ('0' + ((n) >> 28 & 0xF)), \
00670   ('0' + ((n) >> 24 & 0xF)), \
00671   ('0' + ((n) >> 20 & 0xF)), \
00672   ('0' + ((n) >> 16 & 0xF)), \
00673   ('0' + ((n) >> 12 & 0xF)), \
00674   ('0' + ((n) >> 8 & 0xF)), \
00675   ('0' + ((n) >> 4 & 0xF)), \
00676   ('0' + ((n) & 0xF))
00677
00678 /* Construct a string literal encoding the version number. */
00679 #ifdef COMPILER_VERSION
00680 char const* info_version = "INFO" ":" "compiler_version[" COMPILER_VERSION "];"
00681
00682 /* Construct a string literal encoding the version number components. */
00683 #elif defined(COMPILER_VERSION_MAJOR)
00684 char const info_version[] = {
00685   'I', 'N', 'F', 'O', ':',
00686   'c', 'o', 'm', 'p', 'i', 'l', 'e', 'r', '_', 'v', 'e', 'r', 's', 'i', 'o', 'n', '[',
00687   COMPILER_VERSION_MAJOR,
00688   #ifdef COMPILER_VERSION_MINOR
00689   '.', COMPILER_VERSION_MINOR,
00690   #ifdef COMPILER_VERSION_PATCH
00691   '.', COMPILER_VERSION_PATCH,
00692   #ifdef COMPILER_VERSION_TWEAK
00693   '.', COMPILER_VERSION_TWEAK,
00694   #endif
00695   #endif
00696   ']', '\0'};
00697 #endif
00698 #endif
00699
00700 /* Construct a string literal encoding the internal version number. */
00701 #ifdef COMPILER_VERSION_INTERNAL
00702 char const info_version_internal[] = {
00703   'I', 'N', 'F', 'O', ':',
00704   'c', 'o', 'm', 'p', 'i', 'l', 'e', 'r', '_', 'v', 'e', 'r', 's', 'i', 'o', 'n', '_',
00705   'i', 'n', 't', 'e', 'r', 'n', 'a', 'l', '[',
00706   COMPILER_VERSION_INTERNAL, ']', '\0'};
00707 #elif defined(COMPILER_VERSION_INTERNAL_STR)
00708 char const* info_version_internal = "INFO" ":" "compiler_version_internal["
    COMPILER_VERSION_INTERNAL_STR "];"
00709 #endif
00710
00711 /* Construct a string literal encoding the version number components. */
00712 #ifdef SIMULATE_VERSION_MAJOR
00713 char const info_simulate_version[] = {
00714   'I', 'N', 'F', 'O', ':',
00715   's', 'i', 'm', 'u', 'l', 'a', 't', 'e', 'r', '_', 'v', 'e', 'r', 's', 'i', 'o', 'n', '[',
00716   SIMULATE_VERSION_MAJOR,
00717   #ifdef SIMULATE_VERSION_MINOR
00718   '.', SIMULATE_VERSION_MINOR,
00719   #ifdef SIMULATE_VERSION_PATCH
00720   '.', SIMULATE_VERSION_PATCH,
00721   #ifdef SIMULATE_VERSION_TWEAK
00722   '.', SIMULATE_VERSION_TWEAK,
00723   #endif
00724   #endif
00725   ']', '\0'};
00726

```



```

00727 #endif
00728
00729 /* Construct the string literal in pieces to prevent the source from
00730    getting matched. Store it in a pointer rather than an array
00731    because some compilers will just produce instructions to fill the
00732    array rather than assigning a pointer to a static array. */
00733 char const* info_platform = "INFO" ":" "platform[" PLATFORM_ID "];"
00734 char const* info_arch = "INFO" ":" "arch[" ARCHITECTURE_ID "];"
00735
00736
00737
00738 #if defined(__INTEL_COMPILER) && defined(_MSVC_LANG) && _MSVC_LANG < 201403L
00739 #   if defined(__INTEL_CXX11_MODE__)
00740 #       if defined(__cpp_aggregate_nsdmi)
00741 #           define CXX_STD 201402L
00742 #       else
00743 #           define CXX_STD 201103L
00744 #       endif
00745 #   else
00746 #       define CXX_STD 199711L
00747 #   endif
00748 #elif defined(_MSC_VER) && defined(_MSVC_LANG)
00749 #   define CXX_STD _MSVC_LANG
00750 #else
00751 #   define CXX_STD __cplusplus
00752 #endif
00753
00754 const char* info_language_dialect_default = "INFO" ":" "dialect_default["
00755 #if CXX_STD > 202002L
00756     "23"
00757 #elif CXX_STD > 201703L
00758     "20"
00759 #elif CXX_STD >= 201703L
00760     "17"
00761 #elif CXX_STD >= 201402L
00762     "14"
00763 #elif CXX_STD >= 201103L
00764     "11"
00765 #else
00766     "98"
00767 #endif
00768     "];"
00769
00770 /*-----*/
00771
00772 int main(int argc, char* argv[])
00773 {
00774     int require = 0;
00775     require += info_compiler[argc];
00776     require += info_platform[argc];
00777     #ifdef COMPILER_VERSION_MAJOR
00778     require += info_version[argc];
00779     #endif
00780     #ifdef COMPILER_VERSION_INTERNAL
00781     require += info_version_internal[argc];
00782     #endif
00783     #ifdef SIMULATE_ID
00784     require += info_simulate[argc];
00785     #endif
00786     #ifdef SIMULATE_VERSION_MAJOR
00787     require += info_simulate_version[argc];
00788     #endif
00789     #if defined(__CRAYXT_COMPUTE_LINUX_TARGET)
00790     require += info_cray[argc];
00791     #endif
00792     require += info_language_dialect_default[argc];
00793     (void)argv;
00794     return require;
00795 }

```

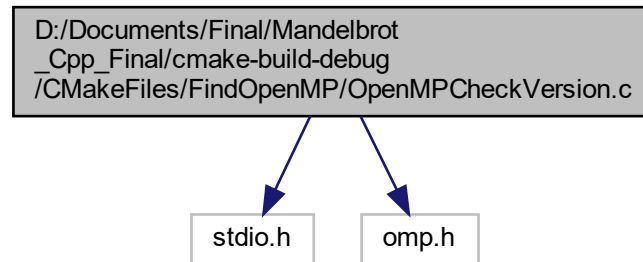
6.5 D:/Documents/Final/Mandelbrot_Cpp_Final/cmake-build-debug/CMakeFiles/FindOpenMP/OpenMPCheckVersion.c File Reference

```

#include <stdio.h>
#include <omp.h>

```

Include dependency graph for `OpenMPCheckVersion.c`:



Functions

- `int main (void)`

Variables

- `const char ompver_str []`

6.5.1 Function Documentation

6.5.1.1 `main()`

```
int main (  
    void )
```

Definition at line 13 of file `OpenMPCheckVersion.c`.

```
00014 {  
00015     puts(ompver_str);  
00016     return 0;  
00017 }
```

6.5.2 Variable Documentation

6.5.2.1 ompver_str

```
const char ompver_str[]
```

Initial value:

```
= { 'I', 'N', 'F', 'O', ':', 'O', 'p', 'e', 'n', 'M',
    'P', '-', 'd', 'a', 't', 'e', '[',
    ('0' + ((_OPENMP/100000)%10)),
    ('0' + ((_OPENMP/10000)%10)),
    ('0' + ((_OPENMP/1000)%10)),
    ('0' + ((_OPENMP/100)%10)),
    ('0' + ((_OPENMP/10)%10)),
    ('0' + ((_OPENMP/1)%10)),
    ']', '\0' }
```

Definition at line 4 of file [OpenMPCheckVersion.c](#).

6.6 OpenMPCheckVersion.c

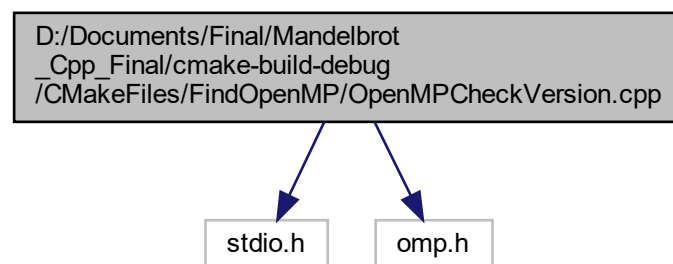
[Go to the documentation of this file.](#)

```
00001
00002 #include <stdio.h>
00003 #include <omp.h>
00004 const char ompver_str[] = { 'I', 'N', 'F', 'O', ':', 'O', 'p', 'e', 'n', 'M',
00005                             'P', '-', 'd', 'a', 't', 'e', '[',
00006                             ('0' + ((_OPENMP/100000)%10)),
00007                             ('0' + ((_OPENMP/10000)%10)),
00008                             ('0' + ((_OPENMP/1000)%10)),
00009                             ('0' + ((_OPENMP/100)%10)),
00010                             ('0' + ((_OPENMP/10)%10)),
00011                             ('0' + ((_OPENMP/1)%10)),
00012                             ']', '\0' };
00013 int main(void)
00014 {
00015     puts(ompver_str);
00016     return 0;
00017 }
```

6.7 D:/Documents/Final/Mandelbrot_Cpp_Final/cmake-build-debug/↵ CMakeFiles/FindOpenMP/OpenMPCheckVersion.cpp File Reference

```
#include <stdio.h>
#include <omp.h>
```

Include dependency graph for OpenMPCheckVersion.cpp:



Functions

- int [main](#) (void)

Variables

- const char [ompver_str](#) []

6.7.1 Function Documentation

6.7.1.1 main()

```
int main (
    void )
```

Definition at line 13 of file [OpenMPCheckVersion.cpp](#).

```
00014 {
00015     puts(ompver_str);
00016     return 0;
00017 }
```

6.7.2 Variable Documentation

6.7.2.1 ompver_str

```
const char ompver_str[]
```

Initial value:

```
= { 'I', 'N', 'F', 'O', ':', 'O', 'p', 'e', 'n', 'M',
    'P', '-', 'd', 'a', 't', 'e', '[',
    ('0' + ((_OPENMP/100000)%10)),
    ('0' + ((_OPENMP/10000)%10)),
    ('0' + ((_OPENMP/1000)%10)),
    ('0' + ((_OPENMP/100)%10)),
    ('0' + ((_OPENMP/10)%10)),
    ('0' + ((_OPENMP/1)%10)),
    ']', '\0' }
```

Definition at line 4 of file [OpenMPCheckVersion.cpp](#).

6.8 OpenMPCheckVersion.cpp

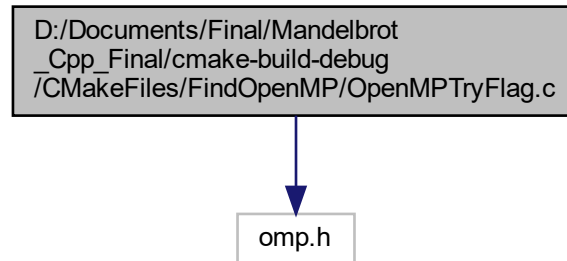
[Go to the documentation of this file.](#)

```
00001
00002 #include <stdio.h>
00003 #include <omp.h>
00004 const char ompver_str[] = { 'I', 'N', 'F', 'O', ':', 'O', 'p', 'e', 'n', 'M',
00005                             'P', '-', 'd', 'a', 't', 'e', '[',
00006                             ('0' + ((_OPENMP/100000)%10)),
00007                             ('0' + ((_OPENMP/10000)%10)),
00008                             ('0' + ((_OPENMP/1000)%10)),
00009                             ('0' + ((_OPENMP/100)%10)),
00010                             ('0' + ((_OPENMP/10)%10)),
00011                             ('0' + ((_OPENMP/1)%10)),
00012                             ']', '\0' };
00013 int main(void)
00014 {
00015     puts(ompver_str);
00016     return 0;
00017 }
```

6.9 D:/Documents/Final/Mandelbrot_Cpp_Final/cmake-build-debug/CMakeFiles/FindOpenMP/OpenMPTryFlag.c File Reference

#include <omp.h>

Include dependency graph for OpenMPTryFlag.c:



Functions

- int [main](#) (void)

6.9.1 Function Documentation

6.9.1.1 main()

```
int main (
    void )
```

Definition at line 3 of file [OpenMPTryFlag.c](#).

```
00003 {
00004 #ifdef _OPENMP
00005     omp_get_max_threads();
00006     return 0;
00007 #elif defined(__HIP_DEVICE_COMPILE__)
00008     return 0;
00009 #else
00010     breaks_on_purpose
00011 #endif
00012 }
```

6.10 OpenMPTryFlag.c

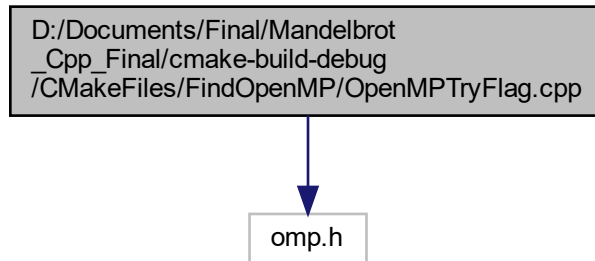
[Go to the documentation of this file.](#)

```
00001
00002 #include <omp.h>
00003 int main(void) {
00004 #ifdef _OPENMP
00005     omp_get_max_threads();
00006     return 0;
00007 #elif defined(__HIP_DEVICE_COMPILE__)
00008     return 0;
00009 #else
00010     breaks_on_purpose
00011 #endif
00012 }
```

6.11 D:/Documents/Final/Mandelbrot_Cpp_Final/cmake-build-debug/CMakeFiles/FindOpenMP/OpenMPTryFlag.cpp File Reference

```
#include <omp.h>
```

Include dependency graph for OpenMPTryFlag.cpp:



Functions

- int [main](#) (void)

6.11.1 Function Documentation

6.11.1.1 main()

```
int main (
    void )
```

Definition at line 3 of file [OpenMPTryFlag.cpp](#).

```
00003 {
00004 #ifdef _OPENMP
00005     omp_get_max_threads();
00006     return 0;
00007 #elif defined(__HIP_DEVICE_COMPILE__)
00008     return 0;
00009 #else
00010     breaks_on_purpose
00011 #endif
00012 }
```

6.12 OpenMPTryFlag.cpp

[Go to the documentation of this file.](#)

```
00001
00002 #include <omp.h>
00003 int main(void) {
00004 #ifdef _OPENMP
00005     omp_get_max_threads();
00006     return 0;
00007 #elif defined(__HIP_DEVICE_COMPILE__)
00008     return 0;
00009 #else
00010     breaks_on_purpose
00011 #endif
00012 }
```

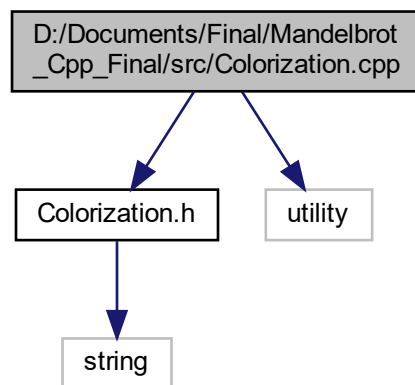
6.13 D:/Documents/Final/Mandelbrot_Cpp_Final/README.md File Reference

6.14 D:/Documents/Final/Mandelbrot_Cpp_Final/src/Colorization.cpp File Reference

```
#include "Colorization.h"
```

```
#include <utility>
```

Include dependency graph for Colorization.cpp:



6.15 Colorization.cpp

[Go to the documentation of this file.](#)

```

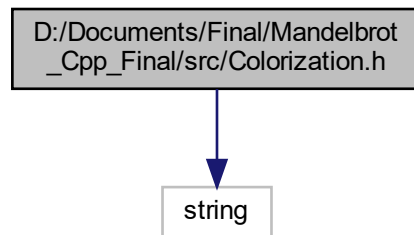
00001 #include "Colorization.h"
00002
00003 #include <utility>
00004
00005 using namespace std;
00006
00007 Colorization::Colorization(string type) : type(std::move(type))
00008 {}
00009
00010 std::string Colorization::get_type()
00011 {
00012     return type;
00013 }
00014
00015 Colorization::~Colorization() = default;

```

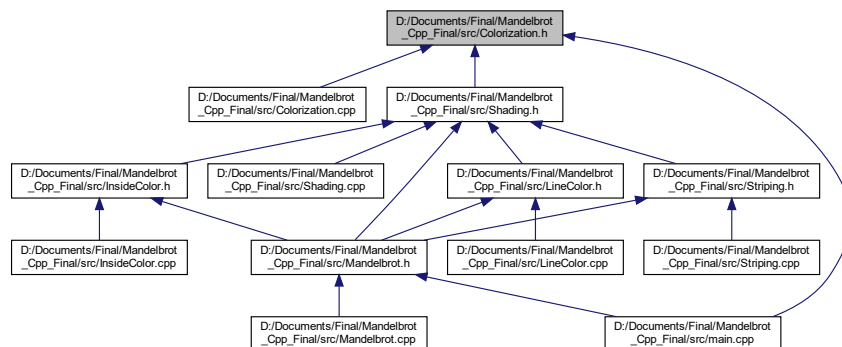
6.16 D:/Documents/Final/Mandelbrot_Cpp_Final/src/Colorization.h File Reference

```
#include <string>
```

Include dependency graph for Colorization.h:



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



Classes

- class [Colorization](#)

6.17 Colorization.h

[Go to the documentation of this file.](#)

```

00001 #ifndef C___COLORIZATION_H
00002 #define C___COLORIZATION_H
00003
00004 #include <string>
00005
00006 class Colorization
00007 {
00008 public:

```



```

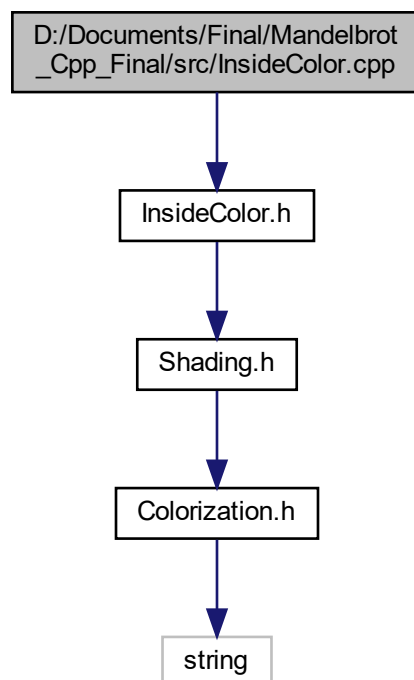
00009
00013     std::string get_type();
00014
00015     virtual unsigned char get_max_color_value() = 0;
00016
00017     virtual unsigned char get_min_color_value() = 0;
00018
00019     virtual ~Colorization();
00020
00021 protected:
00022
00027     explicit Colorization(std::string type);
00028
00032     std::string type;
00033
00034     const unsigned char maxColorValue = 255;
00035
00036     const unsigned char minColorValue = 0;
00037
00038 };
00039
00040 #endif // C__COLORIZATION_H

```

6.18 D:/Documents/Final/Mandelbrot_Cpp_Final/src/InsideColor.cpp File Reference

```
#include "InsideColor.h"
```

Include dependency graph for InsideColor.cpp:



6.19 InsideColor.cpp

[Go to the documentation of this file.](#)

```

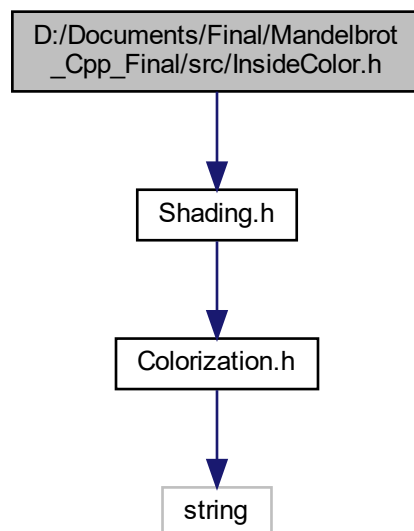
00001 #include "InsideColor.h"
00002
00003 InsideColor::InsideColor() : Shading("Inside")
00004 {}
00005
00006 InsideColor::~InsideColor()
00007 = default;
00008
00009 unsigned char InsideColor::calculate_bw()
00010 {
00011     return minColorValue;
00012 }
00013
00014 unsigned char InsideColor::calculate_r()
00015 {
00016     return minColorValue;
00017 }
00018
00019 unsigned char InsideColor::calculate_g()
00020 {
00021     return minColorValue;
00022 }
00023
00024 unsigned char InsideColor::calculate_b()
00025 {
00026     return minColorValue;
00027 }

```

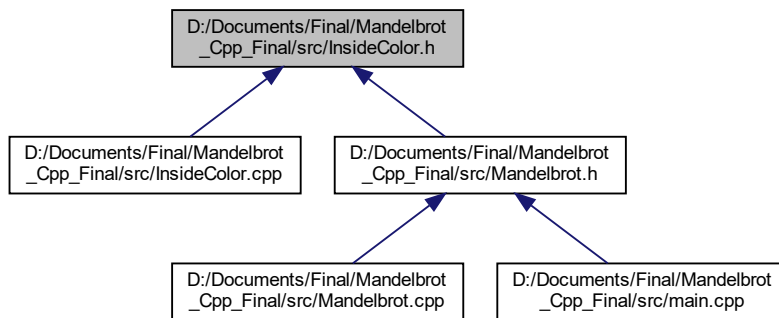
6.20 D:/Documents/Final/Mandelbrot_Cpp_Final/src/InsideColor.h File Reference

#include "Shading.h"

Include dependency graph for InsideColor.h:



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



Classes

- class [InsideColor](#)

6.21 InsideColor.h

[Go to the documentation of this file.](#)

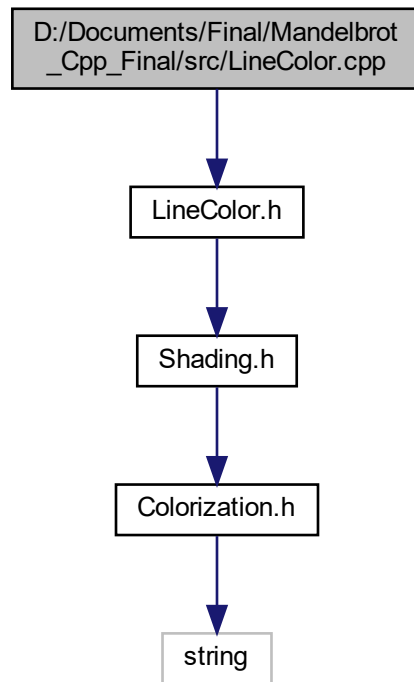
```

00001 #ifndef C___INSIDECOLOR_H
00002 #define C___INSIDECOLOR_H
00003
00004 #include "Shading.h"
00005
00006 class InsideColor : public Shading
00007 {
00008     public:
00009
00010     InsideColor();
00011
00012     ~InsideColor();
00013
00014     unsigned char calculate_bw();
00015
00016     unsigned char calculate_r();
00017
00018     unsigned char calculate_g();
00019
00020     unsigned char calculate_b();
00021 };
00022
00023 #endif //C___INSIDECOLOR_H
  
```

6.22 D:/Documents/Final/Mandelbrot_Cpp_Final/src/LineColor.cpp File Reference

```
#include "LineColor.h"
```

Include dependency graph for LineColor.cpp:



6.23 LineColor.cpp

[Go to the documentation of this file.](#)

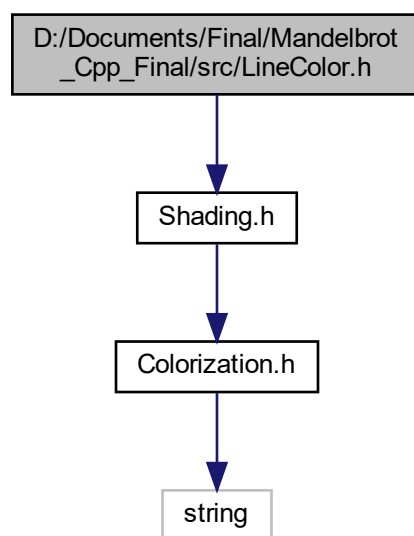
```
00001 #include "LineColor.h"
00002
00003 LineColor::LineColor() : Shading("Line")
00004 {}
00005
00006 LineColor::~~LineColor()
00007 = default;
00008
00009 unsigned char LineColor::calculate_bw()
00010 {
00011     return maxColorValue;
00012 }
00013
00014 unsigned char LineColor::calculate_r()
00015 {
00016     return maxColorValue;
00017 }
00018
00019 unsigned char LineColor::calculate_g()
00020 {
00021     return maxColorValue;
00022 }
00023
```

```
00024 unsigned char LineColor::calculate_b()
00025 {
00026     return maxColorValue;
00027 }
```

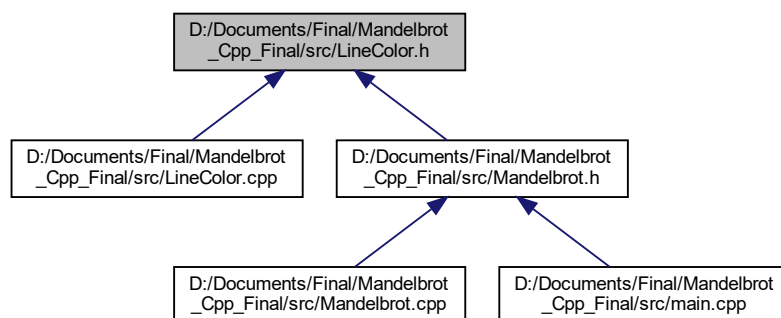
6.24 D:/Documents/Final/Mandelbrot_Cpp_Final/src/LineColor.h File Reference

```
#include "Shading.h"
```

Include dependency graph for LineColor.h:



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



Classes

- class [LineColor](#)

6.25 LineColor.h

[Go to the documentation of this file.](#)

```

00001 #ifndef C___LINECOLOR_H
00002 #define C___LINECOLOR_H
00003
00004 #include "Shading.h"
00005
00006 class LineColor : public Shading
00007 {
00008 public:
00009
00010     LineColor();
00011
00012     ~LineColor();
00013
00014     unsigned char calculate_bw();
00015
00016     unsigned char calculate_r();
00017
00018     unsigned char calculate_g();
00019
00020     unsigned char calculate_b();
00021 };
00022
00023 #endif //C___LINECOLOR_H

```

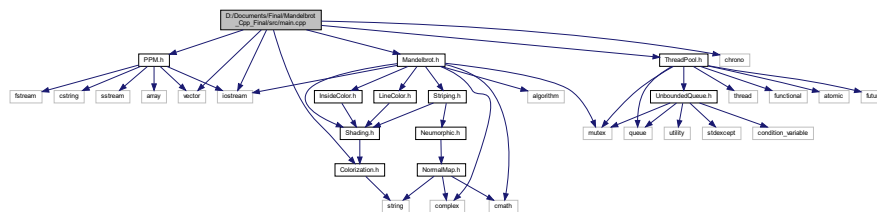
6.26 D:/Documents/Final/Mandelbrot_Cpp_Final/src/main.cpp File Reference

```

#include "PPM.h"
#include "Colorization.h"
#include "Mandelbrot.h"
#include "ThreadPool.h"
#include <chrono>
#include <iostream>
#include <vector>

```

Include dependency graph for main.cpp:



Functions

- int [main](#) ()

6.26.1 Function Documentation

6.26.1.1 main()

```
int main (
    void )
```

Definition at line 12 of file [main.cpp](#).

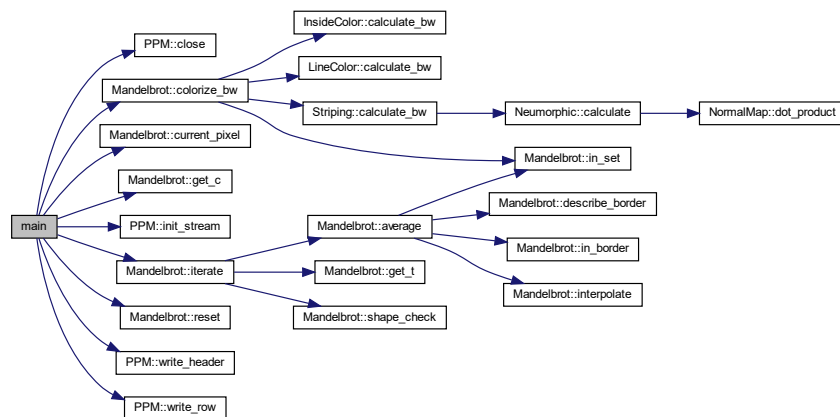
```
00013 {
00014     int width;
00015     int height;
00016
00017     // set up input handling to prevent creation of image with distortion for
00018     // given default Mandelbrot parameters
00019     bool invalidInput = true;
00020     while (invalidInput)
00021     {
00022         cout << "Enter square image size in pixels:\n";
00023         string line;
00024         getline(cin, line);
00025         istringstream is(line);
00026
00027         char dummy;
00028         if (!(is >> width) || (is >> ws && is.get(dummy)) || !(width > 0))
00029         {
00030             cout << "Invalid input. Try again:\n";
00031         } else {
00032             invalidInput = false;
00033         }
00034     }
00035     height = width;
00036
00037
00038     // handle invalid filename inputs
00039     string fileName;
00040     bool invalidChar = true;
00041     while (invalidChar)
00042     {
00043         cout << "Enter output file name in the form 'name.ppm':\n";
00044         cin >> fileName;
00045         for (const char c : fileName)
00046         {
00047             if (!isalnum(c) && !ispunct(c))
00048             {
00049                 cout << "Invalid character found. Try again:\n";
00050                 break;
00051             } else {
00052                 invalidChar = false;
00053                 break;
00054             }
00055         }
00056     }
00057
00058     auto begin = chrono::steady_clock::now();
00059
00060     // set up image stream for writing
00061     PPM pgm(fileName, width, height);
00062     // PPM pgm(width, height); // for testing
00063     if (!pgm.init_stream())
00064     {
00065         cout << "Could not open ofstream for image\n";
00066         return -1;
00067     }
00068     pgm.write_header();
00069
00070     // set up container for image row data
00071     vector<unsigned char> row{}; // array needs compile-time const length
00072     row.resize(width * 3); // avoid constant resizing by allocating up front
00073
00074     cout << "Rendering row by row:\n";
00075
00076     // using basic constructor since I only want this image
00077     // call member functions here to set specific parameters
00078     Mandelbrot gigabrot(width, height);
00079     cout << gigabrot;
00080
00081     // unsigned int numThreads = thread::hardware_concurrency();
```

```

00082 // cout << "numThreads: " << numThreads << "\n";
00083
00084 for (size_t pY = 0; pY < height; pY++)
00085 {
00086     for (size_t pX = 0; pX < width; pX++)
00087     {
00088         size_t subPixel = 3 * pX;
00089         gigabrot.current_pixel(pX, pY);
00090         gigabrot.get_c();
00091         gigabrot.iterate();
00092         row[subPixel + 2] = row[subPixel + 1] = row[subPixel] =
00093             gigabrot.colorize_bw();
00094         gigabrot.reset();
00095     }
00096     {
00097         // implemented due to possibility of having huge image, keep memory usage low
00098         // might be causing the issues with parallelization, ruining the embarrassingly parallel
00099         // aspect of the Mandelbrot set
00100         pgm.write_row(row);
00101     }
00102 }
00103
00104 pgm.close();
00105
00106 auto end = chrono::steady_clock::now();
00107 cout << "Time elapsed: "
00108     << static_cast<float>(chrono::duration_cast<chrono::milliseconds>
00109         (end - begin).count()) / 1000.F
00110     << " sec\n";
00111
00112 return 0;
00113 }

```

Here is the call graph for this function:



6.27 main.cpp

[Go to the documentation of this file.](#)

```

00001 #include "PPM.h"
00002 #include "Colorization.h"
00003 #include "Mandelbrot.h"
00004 #include "ThreadPool.h"
00005
00006 #include <chrono>
00007 #include <iostream>
00008 #include <vector>
00009
00010 using namespace std;
00011
00012 int main()
00013 {
00014     int width;
00015     int height;

```



```

00016
00017 // set up input handling to prevent creation of image with distortion for
00018 // given default Mandelbrot parameters
00019 bool invalidInput = true;
00020 while (invalidInput)
00021 {
00022     cout << "Enter square image size in pixels:\n";
00023     string line;
00024     getline(cin, line);
00025     istringstream is(line);
00026
00027     char dummy;
00028     if (!(is >> width) || (is >> ws && is.get(dummy)) || !(width > 0))
00029     {
00030         cout << "Invalid input. Try again:\n";
00031     } else {
00032         invalidInput = false;
00033     }
00034 }
00035 height = width;
00036
00037
00038 // handle invalid filename inputs
00039 string fileName;
00040 bool invalidChar = true;
00041 while (invalidChar)
00042 {
00043     cout << "Enter output file name in the form 'name.ppm':\n";
00044     cin >> fileName;
00045     for (const char c : fileName)
00046     {
00047         if (!isalnum(c) && !ispunct(c))
00048         {
00049             cout << "Invalid character found. Try again:\n";
00050             break;
00051         } else {
00052             invalidChar = false;
00053             break;
00054         }
00055     }
00056 }
00057
00058 auto begin = chrono::steady_clock::now();
00059
00060 // set up image stream for writing
00061 PPM pgm(fileName, width, height);
00062 // PPM pgm(width, height); // for testing
00063 if (!pgm.init_stream())
00064 {
00065     cout << "Could not open ofstream for image\n";
00066     return -1;
00067 }
00068 pgm.write_header();
00069
00070 // set up container for image row data
00071 vector<unsigned char> row{}; // array needs compile-time const length
00072 row.resize(width * 3); // avoid constant resizing by allocating up front
00073
00074 cout << "Rendering row by row:\n";
00075
00076 // using basic constructor since I only want this image
00077 // call member functions here to set specific parameters
00078 Mandelbrot gigabrot(width, height);
00079 cout << gigabrot;
00080
00081 // unsigned int numThreads = thread::hardware_concurrency();
00082 // cout << "numThreads: " << numThreads << "\n";
00083
00084 for (size_t pY = 0; pY < height; pY++)
00085 {
00086     for (size_t pX = 0; pX < width; pX++)
00087     {
00088         size_t subPixel = 3 * pX;
00089         gigabrot.current_pixel(pX, pY);
00090         gigabrot.get_c();
00091         gigabrot.iterate();
00092         row[subPixel + 2] = row[subPixel + 1] = row[subPixel] =
00093             gigabrot.colorize_bw();
00094         gigabrot.reset();
00095     }
00096     {
00097         // implemented due to possibility of having huge image, keep memory usage low
00098         // might be causing the issues with parallelization, ruining the embarrassingly parallel
00099         // aspect of the Mandelbrot set
00100         pgm.write_row(row);
00101     }
00102 }

```

```

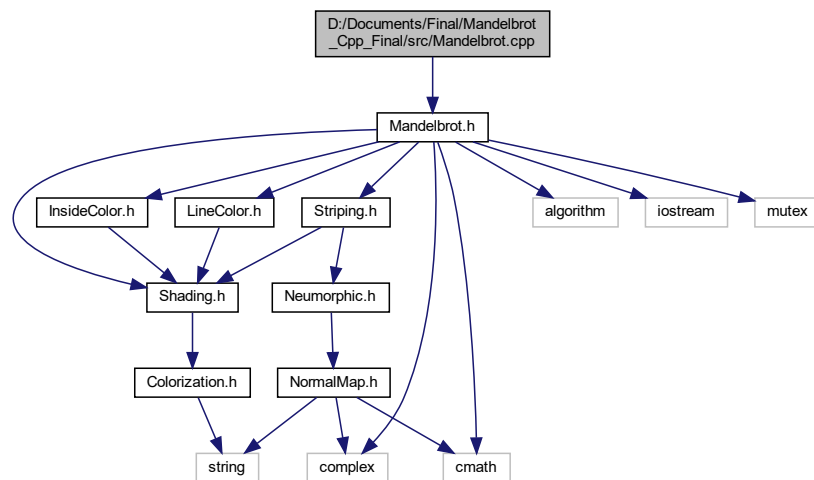
00103
00104     pgm.close();
00105
00106     auto end = chrono::steady_clock::now();
00107     cout << "Time elapsed: "
00108           << static_cast<float>(chrono::duration_cast<chrono::milliseconds>
00109                               (end - begin).count()) / 1000.F
00110           << " sec\n";
00111
00112     return 0;
00113 }

```

6.28 D:/Documents/Final/Mandelbrot_Cpp_Final/src/Mandelbrot.cpp File Reference

```
#include "Mandelbrot.h"
```

Include dependency graph for Mandelbrot.cpp:



Functions

- `std::ostream & operator<< (ostream &os, const Mandelbrot &mandelbrot)`

6.28.1 Function Documentation

6.28.1.1 operator<<()

```

std::ostream & operator<< (
    ostream & os,
    const Mandelbrot & mandelbrot )

```

Definition at line 231 of file [Mandelbrot.cpp](#).

```

00232 {
00233     double pixAspectRatio = (static_cast<double>(mandelbrot.width) / static_cast<double>(mandelbrot
00234         .height));
00235     double worldAspectRatio = (mandelbrot.cxMax - mandelbrot.cxMin) / (mandelbrot.cyMax -
00236         mandelbrot.cyMin);
00237     double distortion = pixAspectRatio - worldAspectRatio;
00238     os << "Distortion (should be 0): " << distortion << "\n";
00239     return os;
00240 }

```

6.29 Mandelbrot.cpp

[Go to the documentation of this file.](#)

```

00001 #include "Mandelbrot.h"
00002
00003 using namespace std;
00004
00005 Mandelbrot::Mandelbrot(int width, int height) : width(width), height(height)
00006 {
00007     pX = 0;
00008     pY = 0;
00009     iter = 0;
00010     iterMax = 1000;
00011     escapeRadius = 1000000.0;
00012     // lnER = log(escapeRadius);
00013     c = 0.0;
00014     r = 0.0;
00015     z = 0.0;
00016     dC = 0.0;
00017     q = 0.0;
00018     cardioid = 0.0;
00019     a = 0.0;
00020     prevA = 0.0;
00021     stripeDensity = 7.0;
00022     d = 0.0;
00023     de = 0.0;
00024     cxMin = -2.2;
00025     cxMax = 0.8;
00026     cyMin = -1.5;
00027     cyMax = 1.5;
00028     pixWidth = 0.0;
00029     pixHeight = 0.0;
00030     iSkip = 1;
00031     thin = 3;
00032     shade = nullptr; // avoid calling "new" more than once per pixel
00033 }
00034
00035 Mandelbrot::Mandelbrot(int pX, int pY, int width, int height) : pX(pX), pY(pY), width(width),
00036 height(height)
00037 {
00038     iter = 0;
00039     iterMax = 1000;
00040     escapeRadius = 1000000.0;
00041     // lnER = log(escapeRadius);
00042     c = 0.0;
00043     r = 0.0;
00044     z = 0.0;
00045     dC = 0.0;
00046     q = 0.0;
00047     cardioid = 0.0;
00048     a = 0.0;
00049     prevA = 0.0;
00050     stripeDensity = 7.0;
00051     d = 0.0;
00052     de = 0.0;
00053     cxMin = -2.2;
00054     cxMax = 0.8;
00055     cyMin = -1.5;
00056     cyMax = 1.5;
00057     pixWidth = 0.0;
00058     pixHeight = 0.0;
00059     iSkip = 1;
00060     thin = 3;
00061     shade = nullptr; // avoid calling "new" more than once per pixel
00062 }
00063
00064 Mandelbrot::~Mandelbrot()
00065 {
00066     delete shade;
00067 }
00068

```

```

00069 void Mandelbrot::set_image(int widthIn, int heightIn)
00070 {
00071     width = widthIn;
00072     height = heightIn;
00073 }
00074
00075 void Mandelbrot::current_pixel(int pxIn, int pyIn)
00076 {
00077     pX = pxIn;
00078     pY = pyIn;
00079 }
00080
00081 void Mandelbrot::set_plane(double cxMinIn, double cxMaxIn, double cyMinIn, double cyMaxIn)
00082 {
00083     cxMin = cxMinIn;
00084     cxMax = cxMaxIn;
00085     cyMin = cyMinIn;
00086     cyMax = cyMaxIn;
00087 }
00088
00089 void Mandelbrot::set_stripe_density(double stripeDensityIn)
00090 {
00091     stripeDensity = stripeDensityIn;
00092 }
00093
00094 void Mandelbrot::set_iSkip(int iSkipIn)
00095 {
00096     iSkip = iSkipIn;
00097 }
00098
00099 void Mandelbrot::set_border(int thinIn)
00100 {
00101     thin = thinIn;
00102 }
00103
00104 void Mandelbrot::get_c()
00105 {
00106     pixWidth = (cxMax-cxMin) / static_cast<double>(width);
00107     pixHeight = (cyMax-cyMin) / static_cast<double>(height);
00108     c = (cxMin + static_cast<double>(pX) * pixWidth) + ((cyMax - static_cast<double>(pY) * pixHeight) *
00109         li);
00109 }
00110
00111 void Mandelbrot::iterate()
00112 {
00113     if (!this->shape_check())
00114     {
00115         for (iter = 0; iter < iterMax; iter++)
00116         {
00117             // mandelbrot set formula
00118             dC = 2.0 * dC * z + 1.0;
00119             z = z * z + c;
00120
00121             // compute average
00122             if (iter > iSkip)
00123             {
00124                 a += get_t();
00125             }
00126
00127             r = abs(z);
00128             if (r > escapeRadius)
00129             {
00130                 break;
00131             }
00132
00133             prevA = a;
00134         }
00135
00136         average();
00137     }
00138 }
00139
00140 unsigned char Mandelbrot::colorize_bw()
00141 {
00142     if (in_set())
00143     {
00144         shade = new InsideColor();
00145         InsideColor *color;
00146         color = dynamic_cast<InsideColor*>(shade);
00147         return color->calculate_bw();
00148     } else if (a == FP_ZERO) {
00149         shade = new LineColor();
00150         LineColor *color;
00151         color = dynamic_cast<LineColor*>(shade);
00152         return color->calculate_bw();
00153     } else {
00154         shade = new Striping(a, z, dC);

```

```

00155     Striping *color;
00156     color = dynamic_cast<Striping*>(shade);
00157     return color->calculate_bw();
00158 }
00159 }
00160
00161 bool Mandelbrot::shape_check()
00162 {
00163     q = ((real(c) - 0.25) * (real(c) - 0.25)) + (imag(c) * imag(c));
00164     cardioid = 0.25 * imag(c) * imag(c);
00165     if ((real(c) * real(c) + 2.0 * real(c) + 1.0 + imag(c) * imag(c)) < bulb ||
00166         (q * (q + (real(c) - 0.25)) < cardioid))
00167     {
00168         return true;
00169     } else {
00170         return false;
00171     }
00172 }
00173
00174 double Mandelbrot::get_t()
00175 {
00176     return 0.5 + 0.5 * sin(stripeDensity * arg(z));
00177 }
00178
00179 void Mandelbrot::interpolate()
00180 {
00181     // smooth iteration count
00182     d = static_cast<double>(iter + 1) + log(log(escapeRadius) / log(r)) / M_LN2;
00183     d = d - static_cast<double>(static_cast<int>(d)); // only fractional part = interpolation
00184     // coefficient
00185 }
00186
00187 void Mandelbrot::average()
00188 {
00189     if (in_set())
00190     {
00191         a = -1.0;
00192     } else {
00193         describe_border();
00194         if (in_border()) // in border
00195         {
00196             a = FP_ZERO;
00197         } else {
00198             a /= static_cast<double>((iter - iSkip)); // A(n)
00199             prevA /= static_cast<double>((iter - iSkip - 1)); // A(n-1)
00200             this->interpolate();
00201             a = (d * a) + ((1.0 - d) * prevA);
00202         }
00203     }
00204 }
00205
00206 void Mandelbrot::describe_border()
00207 {
00208     de = 2.0 * r * log(r) / abs(dC);
00209 }
00210
00211 bool Mandelbrot::in_border()
00212 {
00213     if (de < (pixWidth / static_cast<double>(thin)))
00214     {
00215         return true;
00216     } else {
00217         return false;
00218     }
00219 }
00220
00221 bool Mandelbrot::in_set()
00222 {
00223     if (iter == iterMax)
00224     {
00225         return true;
00226     } else {
00227         return false;
00228     }
00229 }
00230
00231 std::ostream &operator<<(ostream &os, const Mandelbrot& mandelbrot)
00232 {
00233     double pixAspectRatio = (static_cast<double>(mandelbrot.width) / static_cast<double>(mandelbrot
00234     .height));
00235     double worldAspectRatio = (mandelbrot.cxMax - mandelbrot.cxMin) / (mandelbrot.cyMax -
00236     mandelbrot.cyMin);
00237     double distortion = pixAspectRatio - worldAspectRatio;
00238     os << "Distortion (should be 0): " << distortion << "\n";
00239     return os;
00240 }
00241

```

```

00242 void Mandelbrot::reset()
00243 {
00244     c = 0.0;
00245     r = 0.0;
00246     z = 0.0;
00247     dC = 0.0;
00248     q = 0.0;
00249     cardioid = 0.0;
00250     a = 0.0;
00251     prevA = 0.0;
00252     d = 0.0;
00253     shade = nullptr; // avoid calling "new" more than once per pixel
00254 }
00255
00256 Mandelbrot::Mandelbrot(const Mandelbrot &oldMandelbrot) : Mandelbrot(oldMandelbrot.width,
00257                                                                    oldMandelbrot.height)
00258 {}
00259
00260

```

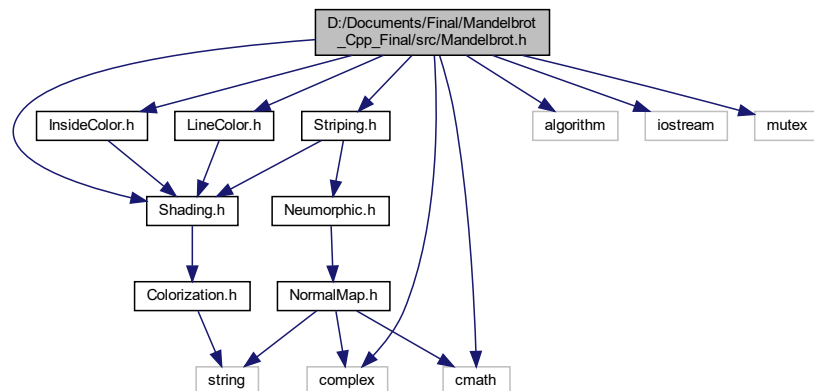
6.30 D:/Documents/Final/Mandelbrot_Cpp_Final/src/Mandelbrot.h File Reference

```

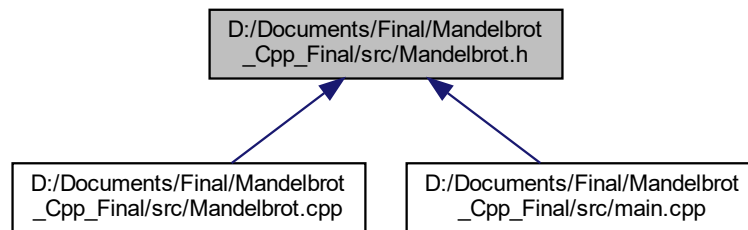
#include "Shading.h"
#include "InsideColor.h"
#include "LineColor.h"
#include "Striping.h"
#include <cmath>
#include <complex>
#include <algorithm>
#include <iostream>
#include <mutex>

```

Include dependency graph for Mandelbrot.h:



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



Classes

- class [Mandelbrot](#)

Macros

- #define [M_PI](#) 3.14159265358979323846

6.30.1 Macro Definition Documentation

6.30.1.1 M_PI

```
#define M_PI 3.14159265358979323846
```

Definition at line 14 of file [Mandelbrot.h](#).

6.31 Mandelbrot.h

[Go to the documentation of this file.](#)

```

00001 #ifndef C__MANDELBROT_H
00002 #define C__MANDELBROT_H
00003
00004 #include "Shading.h"
00005 #include "InsideColor.h"
00006 #include "LineColor.h"
00007 #include "Striping.h"
00008 #include <cmath>
00009 #include <complex>
00010 #include <algorithm>
00011 #include <iostream>
00012 #include <mutex>
00013
00014 #define M_PI 3.14159265358979323846
00015
00016 class Mandelbrot
00017 {

```

```

00018 public:
00019
00025     Mandelbrot(int width, int height);
00026
00035     Mandelbrot(int pX, int pY, int width, int height);
00036
00037     Mandelbrot(const Mandelbrot &oldMandelbrot);
00038
00039     ~Mandelbrot();
00040
00046     void current_pixel(int pxIn, int pyIn);
00047
00048     void set_image(int widthIn, int heightIn);
00049
00050     void set_plane(double cxMinIn, double cxMaxIn, double cyMinIn, double cyMaxIn);
00051
00052     void set_stripe_density(double stripeDensityIn);
00053
00054     void set_iSkip(int iSkipIn);
00055
00056     void set_border(int thinIn);
00057
00061     void get_c();
00062
00066     void iterate();
00067
00071     unsigned char colorize_bw();
00072
00080     bool shape_check();
00081
00086     double get_t();
00087
00091     void interpolate();
00092
00093     void average();
00094
00095     void describe_border();
00096
00097     bool in_border();
00098
00099     bool in_set(); // for readability
00100
00101     friend std::ostream& operator<<(std::ostream& os, const Mandelbrot& mandelbrot);
00102
00103     void reset();
00104
00105 private:
00106
00107     int iter;
00108
00109     int iterMax;
00110
00111     double escapeRadius;
00112
00113     // double lnER;
00114
00115     std::complex<double> c;
00116
00117     double r;
00118
00119     std::complex<double> z;
00120
00121     std::complex<double> dC;
00122
00123     double q;
00124
00125     double cardioid;
00126
00127     const double bulb = 0.0625;
00128
00129     // coordinate plane
00130     double cxMin;
00131
00132     double cxMax;
00133
00134     double cyMin;
00135
00136     double cyMax;
00137
00138     // image
00139     int width;
00140
00141     int height;
00142
00143     int pX;
00144
00145

```



```

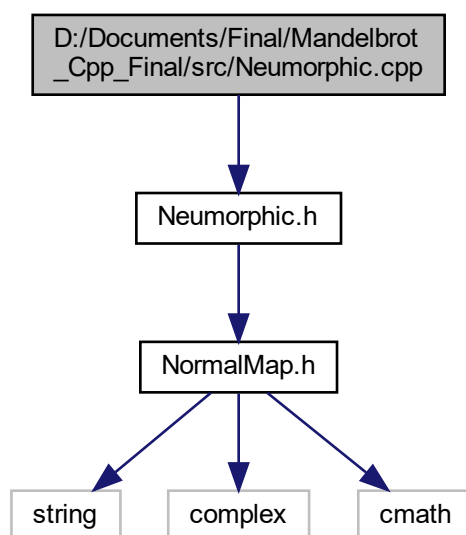
00149  int pY;
00150
00151  double pixWidth;
00152
00153  double pixHeight;
00154
00155  // average
00156  double a;
00157
00158  double prevA;
00159
00163  double stripeDensity;
00164
00168  int iSkip;
00169
00170  // interpolated
00171  double d;
00172
00173  // boundary descriptor
00174  double de;
00175  int thin;
00176
00177  Shading *shade;
00178 };
00179
00180 #endif //C___MANDELBROT_H

```

6.32 D:/Documents/Final/Mandelbrot_Cpp_Final/src/Neumorphic.cpp File Reference

```
#include "Neumorphic.h"
```

Include dependency graph for Neumorphic.cpp:



6.33 Neumorphic.cpp

[Go to the documentation of this file.](#)

```

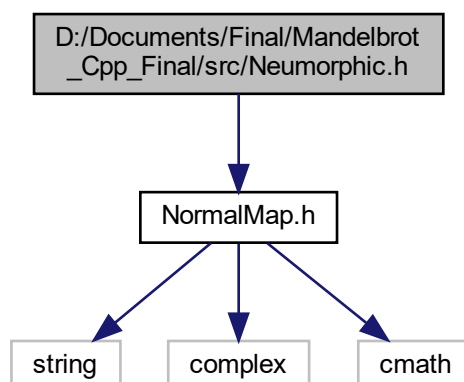
00001 #include "Neumorphic.h"
00002
00003 using namespace std;
00004
00005 Neumorphic::Neumorphic(complex<double> z, complex<double> dC) : NormalMap("Neumorphic"), z(z), dC
00006 (dC)
00007 {
00008     heightFactor = 1.5;
00009     angle = 45.0 / 360.0;
00010     reflection = FP_ZERO;
00011     v = exp(2.0 * angle * M_PI * 1i);
00012 }
00013
00014 double Neumorphic::calculate()
00015 {
00016     u = z / dC;
00017     u = u / abs(u); // normalize
00018     reflection = dot_product(u, v) + heightFactor;
00019     reflection = reflection / (1.0 + heightFactor); // rescale so that it does not get bigger than 1
00020     if (reflection < 0.0)
00021     {
00022         reflection = 0.0;
00023     } else {}
00024     return reflection;
00025 }
00026
00027 double Neumorphic::get_reflection()
00028 {
00029     return reflection;
00030 }
00031
00032 double Neumorphic::get_heightFactor()
00033 {
00034     return heightFactor;
00035 }
00036
00037 double Neumorphic::get_angle()
00038 {
00039     return angle;
00040 }

```

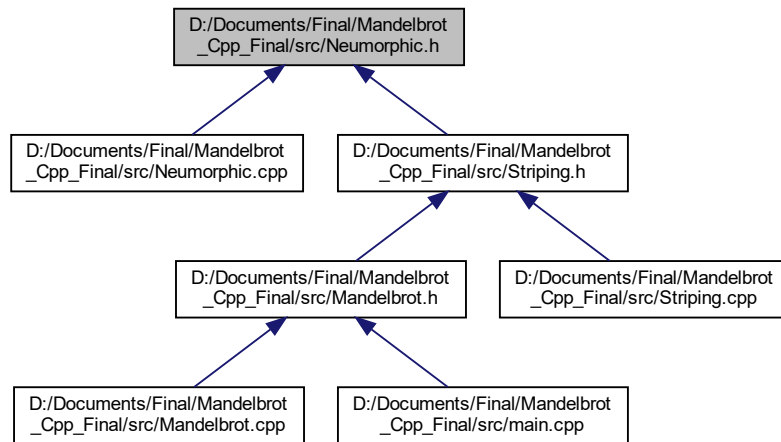
6.34 D:/Documents/Final/Mandelbrot_Cpp_Final/src/Neumorphic.h File Reference

#include "NormalMap.h"

Include dependency graph for Neumorphic.h:



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



Classes

- class [Neumorphic](#)

6.35 Neumorphic.h

[Go to the documentation of this file.](#)

```

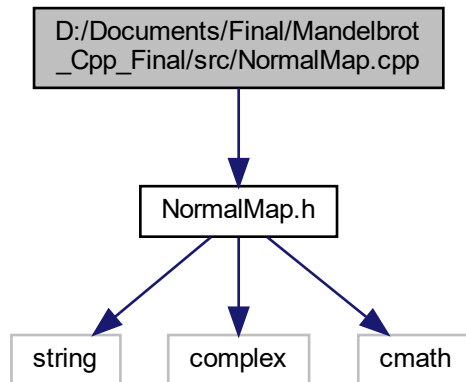
00001 #ifndef C___NEUMORPHIC_H_
00002 #define C___NEUMORPHIC_H_
00003
00004 #include "NormalMap.h"
00005
00006 class Neumorphic : public NormalMap
00007 {
00008 public:
00009
00015     Neumorphic(std::complex<double> z, std::complex<double> dC);
00016
00017     double calculate();
00018
00019     double get_reflection();
00020
00021     double get_heightFactor();
00022
00023     double get_angle();
00024 private:
00025
00026
00027     std::complex<double> z;
00028
00029     std::complex<double> dC;
00030
00034     double heightFactor;
00035
00040     double angle;
00041
00045     double reflection;
00046
00047     std::complex<double> u;
00048
00052     std::complex<double> v;
00053
00054 };
00055
00056 #endif //C___NEUMORPHIC_H_

```

6.36 D:/Documents/Final/Mandelbrot_Cpp_Final/src/NormalMap.cpp File Reference

```
#include "NormalMap.h"
```

Include dependency graph for NormalMap.cpp:



6.37 NormalMap.cpp

[Go to the documentation of this file.](#)

```

00001 #include "NormalMap.h"
00002
00003 using namespace std;
00004
00005 NormalMap::NormalMap(string type)
00006 {}
00007
00008 NormalMap::~NormalMap()
00009 = default;
00010
00011 double NormalMap::dot_product(std::complex<double> u, std::complex<double> v)
00012 {
00013     return real(u) * real(v) + imag(u) * imag(v);
00014 }
00015
00016 double NormalMap::get_min_val()
00017 {
00018     return minMapVal;
00019 }
00020
00021 double NormalMap::get_max_val()
00022 {
00023     return maxMapVal;
00024 }
00025 std::string NormalMap::get_type()
00026 {
00027     return type;
00028 }

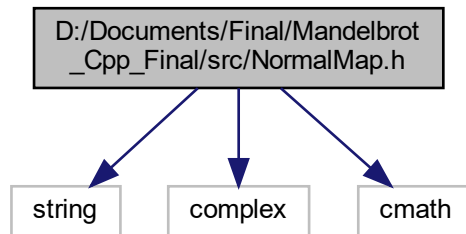
```

6.38 D:/Documents/Final/Mandelbrot_Cpp_Final/src/NormalMap.h File Reference

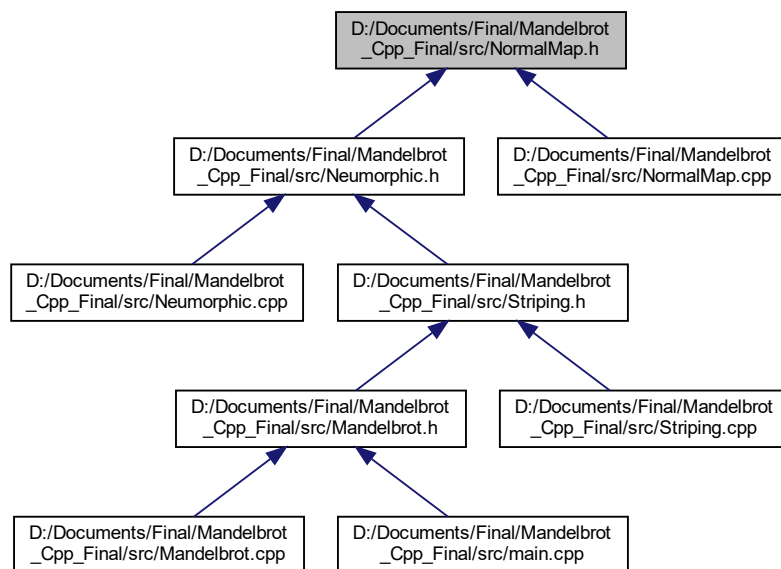
```
#include <string>
#include <complex>
```

```
#include <cmath>
```

Include dependency graph for NormalMap.h:



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



Classes

- class [NormalMap](#)

6.39 NormalMap.h

[Go to the documentation of this file.](#)

```
00001 #ifndef C___NORMALMAP_H
00002 #define C___NORMALMAP_H
```

```

00003
00004 #include <string>
00005 #include <complex>
00006 #include <cmath>
00007
00008 class NormalMap
00009 {
00010 public:
00011
00012 ~NormalMap();
00013
00014 virtual double calculate() = 0;
00015
00022 double dot_product(std::complex<double> u, std::complex<double> v);
00023
00024 double get_min_val();
00025
00026 double get_max_val();
00027
00028 std::string get_type();
00029
00030 protected:
00031
00032 explicit NormalMap(std::string type);
00033
00034 private:
00035
00036 const double minMapVal = 0.0;
00037
00038 const double maxMapVal = 1.0;
00039
00040 std::string type;
00041 };
00042
00043 #endif //C___NORMALMAP_H

```

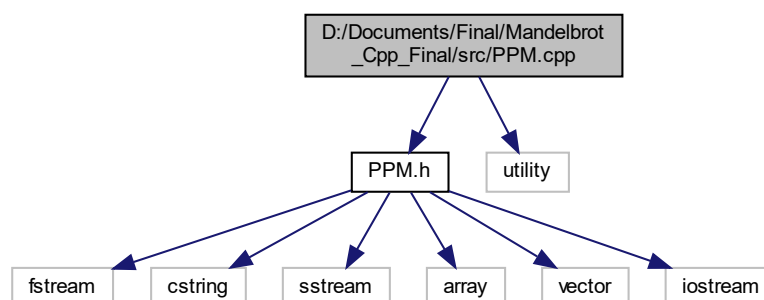
6.40 D:/Documents/Final/Mandelbrot_Cpp_Final/src/PPM.cpp File Reference

```

#include "PPM.h"
#include <utility>

```

Include dependency graph for PPM.cpp:



6.41 PPM.cpp

[Go to the documentation of this file.](#)

```

00001 #include "PPM.h"
00002

```

```

00003 #include <utility>
00004
00005 using namespace std;
00006
00007 PPM::PPM(int width, int height) : width(width), height(height)
00008 {
00009     outputDirectory = "..\\output\\";
00010     fileName = outputDirectory + "gigabrot_default.ppm";
00011     subPixel = width * 3;
00012 }
00013
00014 PPM::PPM(const std::string &fileName, int width, int height) : width(width), height(height)
00015 {
00016     outputDirectory = "..\\output\\";
00017     this->fileName = outputDirectory + fileName;
00018     subPixel = width * 3;
00019 }
00020
00021 PPM::PPM(const PPM& oldPPM) : magic(oldPPM.magic), pixMaxVal(oldPPM.pixMaxVal),
00022     width(oldPPM.width), height(oldPPM.height), subPixel(oldPPM.subPixel), comment(oldPPM.comment)
00023 {}
00024
00025 void PPM::set_outputDirectory(const std::string &outputDirectoryIn)
00026 {
00027     outputDirectory = outputDirectoryIn;
00028 }
00029
00030 PPM& PPM::operator=(string fileNameIn)
00031 {
00032     this->fileName = std::move(fileNameIn);
00033     return *this;
00034 }
00035
00036 bool PPM::init_stream()
00037 {
00038     image.open(fileName, ios::binary);
00039
00040     if (image.is_open())
00041     {
00042         return true;
00043     } else
00044     {
00045         return false;
00046     }
00047 }
00048
00049 void PPM::write_header()
00050 {
00051     string widthStr = to_string(this->width);
00052     string lengthStr = to_string(this->height);
00053     header << magic << widthStr << " " << lengthStr << "\n" << comment << "\n" << pixMaxVal;
00054     image << header.rdbuf();
00055 }
00056
00057 void PPM::set_width(int widthIn)
00058 {
00059     width = widthIn;
00060 }
00061
00062 void PPM::set_height(int heightIn)
00063 {
00064     height = heightIn;
00065 }
00066
00067 void PPM::set_comment(string commentIn)
00068 {
00069     comment = std::move(commentIn);
00070 }
00071
00072 void PPM::close()
00073 {
00074     image.close();
00075     cout << "File " << fileName << " saved\n";
00076 }

```

6.42 D:/Documents/Final/Mandelbrot_Cpp_Final/src/PPM.h File Reference

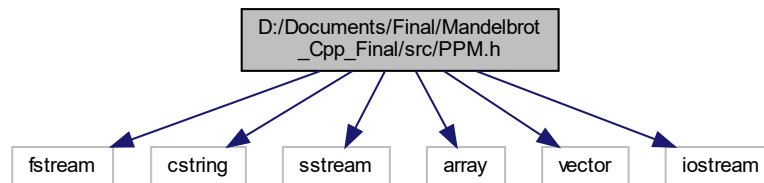
```

#include <fstream>
#include <cstring>

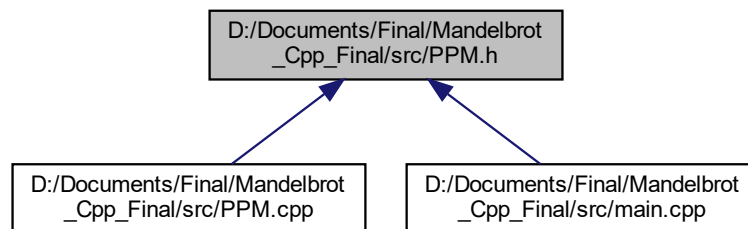
```

```
#include <sstream>
#include <array>
#include <vector>
#include <iostream>
```

Include dependency graph for PPM.h:



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



Classes

- class [PPM](#)

6.43 PPM.h

[Go to the documentation of this file.](#)

```
00001 #ifndef C___PPM_H
00002 #define C___PPM_H
00003
00004 #include <fstream>
00005 #include <cstring>
00006 #include <sstream>
00007 #include <array>
00008 #include <vector>
00009 #include <iostream>
00010
00011 class PPM
00012 {
00013 public:
00014
00021     PPM(int width, int height);
00022
```



```

00030     PPM(const std::string& fileName, int width, int height);
00031
00036     PPM(const PPM& oldPPM);
00037
00042     void set_outputDirectory(const std::string& outputDirectoryIn);
00043
00049     PPM& operator=(const std::string fileNameIn);
00050
00055     bool init_stream();
00056
00062     //template<size_t N>
00063     void write_row(const std::vector<unsigned char> &row)
00064     {
00065         image.write((char const *) row.data(), row.size());
00066     }
00067
00071     void write_header();
00072
00077     void set_width(int widthIn);
00078
00083     void set_height(int heightIn);
00084
00089     void set_comment(std::string commentIn);
00090
00094     void close();
00095
00096
00097 private:
00098
00102     const std::string magic = "P6\n";
00103
00104     const std::string pixMaxVal = "255\n";
00105
00106     int width;
00107
00108     int subPixel;
00109
00110     int height;
00111
00112     std::string comment;
00113
00117     std::stringstream header;
00118
00119     std::string outputDirectory;
00120
00121     std::string fileName;
00122
00123     std::ofstream image;
00124 };
00125
00126 #endif //C___PPM_H

```

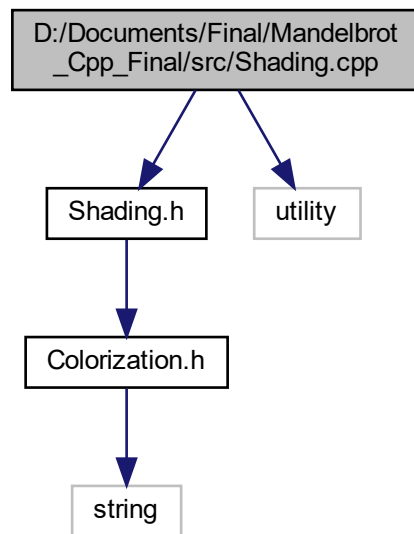
6.44 D:/Documents/Final/Mandelbrot_Cpp_Final/src/Shading.cpp File Reference

```

#include "Shading.h"
#include <utility>

```

Include dependency graph for Shading.cpp:

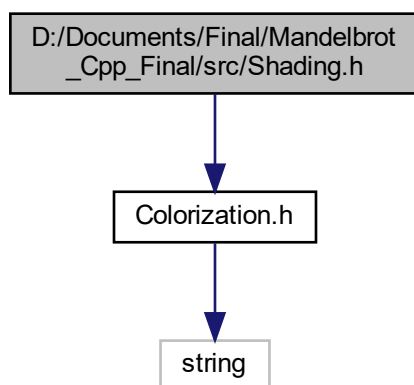


6.45 Shading.cpp

[Go to the documentation of this file.](#)

```
00001 #include "Shading.h"
00002
00003 #include <utility>
00004
00005 Shading::Shading(std::string type) : Colorization(std::move(type))
00006 {}
00007
00008 Shading::~Shading()
00009 = default;
00010
00011 unsigned char Shading::get_max_color_value()
00012 {
00013     return maxColorValue;
00014 }
00015
00016 unsigned char Shading::get_min_color_value()
00017 {
00018     return minColorValue;
00019 }
```

```
#include "Colorization.h"
Include dependency graph for Shading.h:
```



```

graph TD
    A["D:\Documents\Final\Mandelbrot_Cpp_Final\src\Shading.h"] --> B["D:\Documents\Final\Mandelbrot_Cpp_Final\src\InsideColor.h"]
    A --> C["D:\Documents\Final\Mandelbrot_Cpp_Final\src\LineColor.h"]
    A --> D["D:\Documents\Final\Mandelbrot_Cpp_Final\src\Striping.h"]
    A --> E["D:\Documents\Final\Mandelbrot_Cpp_Final\src\Shading.cpp"]
    B --> F["D:\Documents\Final\Mandelbrot_Cpp_Final\src\InsideColor.cpp"]
    B --> G["D:\Documents\Final\Mandelbrot_Cpp_Final\src\Mandelbrot.h"]
    C --> H["D:\Documents\Final\Mandelbrot_Cpp_Final\src\LineColor.cpp"]
    C --> G
    D --> I["D:\Documents\Final\Mandelbrot_Cpp_Final\src\Striping.cpp"]
    D --> G
    E --> G
    G --> J["D:\Documents\Final\Mandelbrot_Cpp_Final\src\Mandelbrot.cpp"]
    G --> K["D:\Documents\Final\Mandelbrot_Cpp_Final\src\main.cpp"]
  
```

- class **Shading**

[Go to the documentation of this file.](#)

```
00001 #ifndef C__SHADING_H
00002 #define C__SHADING_H
00003
00004 #include "Colorization.h"
00005
00006 class Shading : public Colorization
```

```

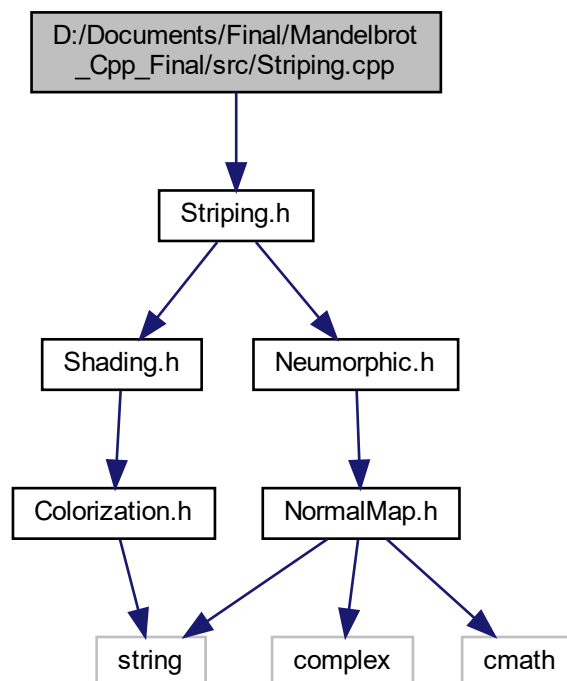
00007 {
00008     public:
00009         explicit Shading(std::string subtype);
00010         ~Shading();
00011         virtual unsigned char get_max_color_value();
00012         virtual unsigned char get_min_color_value();
00013         virtual unsigned char calculate_bw() = 0;
00014         // option for other colorizations
00015         virtual unsigned char calculate_r() = 0;
00016         virtual unsigned char calculate_g() = 0;
00017         virtual unsigned char calculate_b() = 0;
00018     private:
00019         std::string type = "Shading";
00020 };
00021 #endif // C___SHADING_H

```

6.48 D:/Documents/Final/Mandelbrot_Cpp_Final/src/Striping.cpp File Reference

```
#include "Striping.h"
```

Include dependency graph for Striping.cpp:



6.49 Striping.cpp

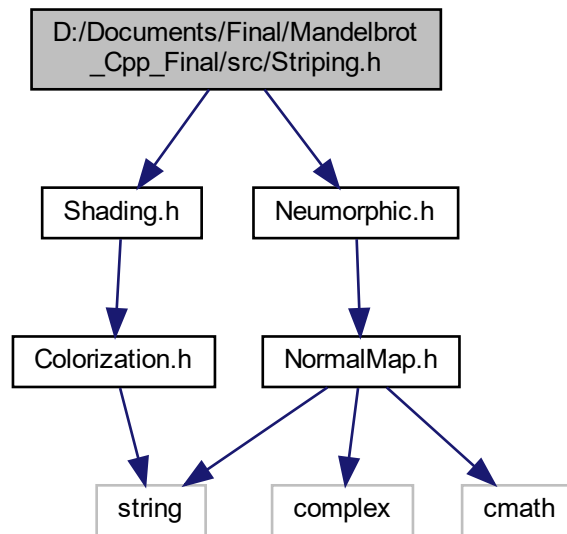
[Go to the documentation of this file.](#)

```
00001 #include "Striping.h"
00002
00003 using namespace std;
00004
00005 Striping::Striping(double average, complex<double> z, complex<double> dc) : Shading("Striping"),
00006 average(average), reflection(z, dc)
00007 {}
00008
00009 Striping::~Striping()
00010 = default;
00011
00012 unsigned char Striping::calculate_bw()
00013 {
00014     return static_cast<unsigned char>((static_cast<double>((maxColorValue - 1)) - (100.0 *
00015     average)) * reflection.calculate()); // explicit casting
00016 }
00017
00018 unsigned char Striping::calculate_r()
00019 {
00020     return 0;
00021 }
00022
00023 unsigned char Striping::calculate_g()
00024 {
00025     return 0;
00026 }
00027
00028 unsigned char Striping::calculate_b()
00029 {
00030     return 0;
00031 }
00032
00033 double Striping::get_average(double a)
00034 {
00035     return average;
00036 }
```

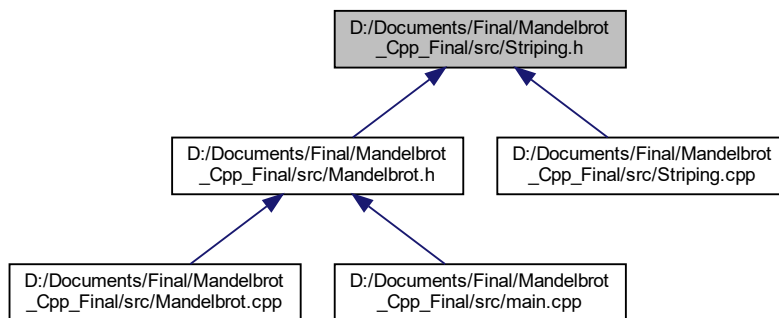
6.50 D:/Documents/Final/Mandelbrot_Cpp_Final/src/Striping.h File Reference

```
#include "Shading.h"
#include "Neumorphic.h"
```

Include dependency graph for Striping.h:



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



Classes

- class [Striping](#)

6.51 Striping.h

[Go to the documentation of this file.](#)

```
00001 #ifndef C___STRIPING_H
```

```

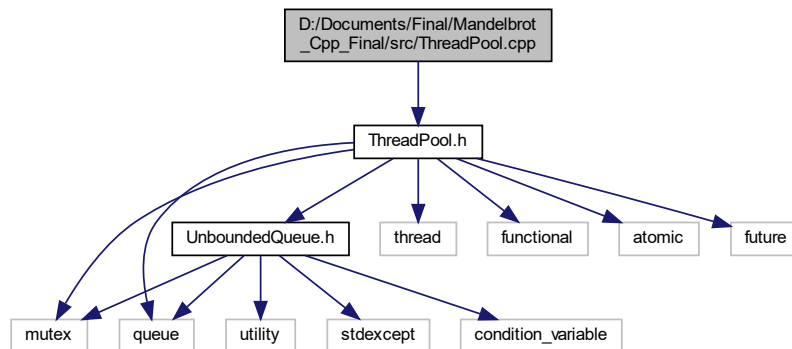
00002 #define C___STRIPING_H
00003
00004 #include "Shading.h"
00005 #include "Neumorphic.h"
00006
00007 class Striping : public Shading
00008 {
00009 public:
00010
00011     Striping(double average, std::complex<double> z, std::complex<double> dc);
00012
00013     ~Striping();
00014
00015     unsigned char calculate_bw();
00016
00017     unsigned char calculate_r();
00018
00019     unsigned char calculate_g();
00020
00021     unsigned char calculate_b();
00022
00023     double get_average(double average);
00024
00025 private:
00026
00027     double average;
00028
00029     Neumorphic reflection;
00030 };
00031
00032 #endif //C___STRIPING_H

```

6.52 D:/Documents/Final/Mandelbrot_Cpp_Final/src/ThreadPool.cpp File Reference

```
#include "ThreadPool.h"
```

Include dependency graph for ThreadPool.cpp:



6.53 ThreadPool.cpp

[Go to the documentation of this file.](#)

```

00001 #include "ThreadPool.h"
00002
00003 ThreadPool::ThreadPool(unsigned int numThreads) : queues(numThreads), count(numThreads)
00004 {
00005     if (!numThreads)
00006     {
00007         throw std::invalid_argument("thread count must be nonzero!\n");

```

```

00008     } else if (numThreads < 0) {
00009         throw std::invalid_argument("thread count must be positive! how did this happen??");
00010     }
00011
00012     auto worker = [this] (auto i) {
00013         while (true)
00014         {
00015             process proc;
00016             for (auto j = 0; j < count * countMult; j++)
00017             {
00018                 if (queues[(i + j) % count].try_pop(proc))
00019                 {
00020                     break;
00021                 }
00022             }
00023             if (!proc && !queues[i].pop(proc))
00024             {
00025                 break;
00026             }
00027             proc();
00028         }
00029     };
00030
00031     threads.reserve(numThreads);
00032
00033     for (auto i = 0; i < numThreads; i++)
00034     {
00035         threads.emplace_back(worker, i);
00036     }
00037 }
00038
00039 ThreadPool::~ThreadPool()
00040 {
00041     for (auto& queue: queues)
00042     {
00043         queue.unblock();
00044     }
00045     for (auto& thread : threads)
00046     {
00047         thread.join();
00048     }
00049 }

```

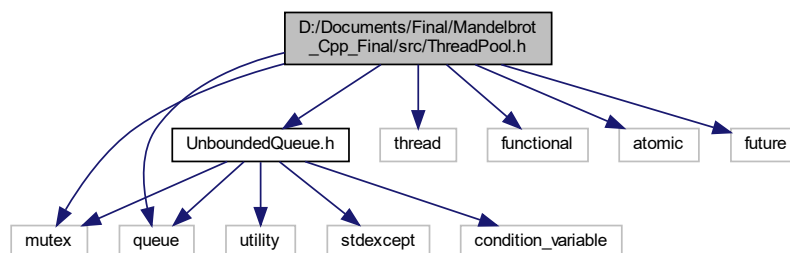
6.54 D:/Documents/Final/Mandelbrot_Cpp_Final/src/ThreadPool.h File Reference

```

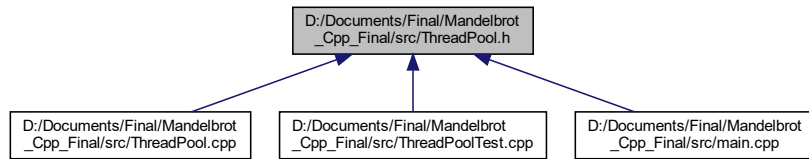
#include "UnboundedQueue.h"
#include <thread>
#include <mutex>
#include <queue>
#include <functional>
#include <atomic>
#include <future>

```

Include dependency graph for ThreadPool.h:



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



Classes

- class [ThreadPool](#)

6.55 ThreadPool.h

[Go to the documentation of this file.](#)

```

00001 #ifndef C___THREADPOOL_H_
00002 #define C___THREADPOOL_H_
00003
00004 #include "UnboundedQueue.h"
00005
00006 #include <thread>
00007 #include <mutex>
00008 #include <queue>
00009 #include <functional>
00010 #include <atomic>
00011 #include <future>
00012
00016 class ThreadPool
00017 {
00018 public:
00019
00020     explicit ThreadPool(unsigned int numThreads);
00021
00022     ~ThreadPool();
00023
00024     template<typename T, typename... ARGS>
00025     void enqueue_work(T&& t, ARGS&&... args)
00026     {
00027         auto work = [proc = std::forward<T>(t), tuple = std::make_tuple(std::forward<ARGS>(args)...)]
00028             () {std::apply(proc, tuple);};
00029
00030         auto i = index++;
00031
00032         for (auto j = 0; j < count * countMult; j++)
00033         {
00034             if (queues[(i + j) % count].try_push(work))
00035             {
00036                 return;
00037             }
00038         }
00039
00040         queues[i % count].push(std::move(work));
00041     }
00042
00043     template<typename T, typename... ARGS>
00044     void enqueue_task(T&& t, ARGS&&... args)
00045     {
00046         using taskReturnType = std::invoke_result<T, ARGS...>;
00047         using taskType = std::packaged_task<taskReturnType>;
00048
00049         auto task = std::make_shared<taskType>(std::bind(std::forward<T>(t),
00050             std::forward<ARGS>(args)...));
00051         auto work = [=] () {(*task)();};
00052         auto result = task->get_future();
00053         auto i = index++;
00054     }

```

```

00055     for (auto j = 0; j < count * countMult; j++)
00056     {
00057         if (queues[(i + j) % count].try_push(work))
00058         {
00059             return result;
00060         }
00061     }
00062     queues[i % count].push(std::move(work));
00063     return result;
00064 }
00065 private:
00066 using process = std::function<void(void)>;
00067 using queue = UnboundedQueue<process>;
00068 using queueVec = std::vector<queue>;
00069 queueVec queues;
00070 using Threads = std::vector<std::thread>;
00071 Threads threads;
00072 const std::size_t count;
00073 std::atomic_uint index = 0;
00074 inline static const unsigned int countMult = 2;
00075 };
00076 #endif //C_____THREADPOOL_H_

```

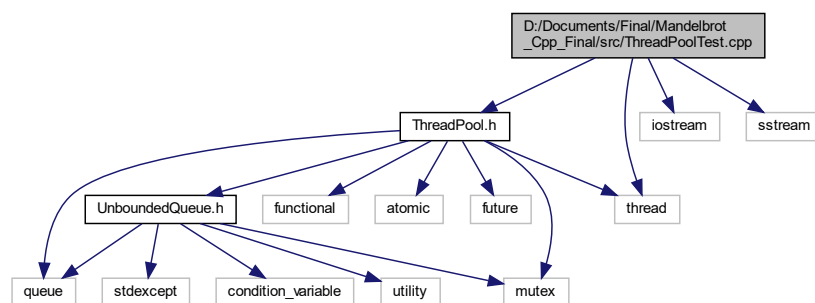
6.56 D:/Documents/Final/Mandelbrot_Cpp_Final/src/ThreadPoolTest.cpp File Reference

```

#include "ThreadPool.h"
#include <thread>
#include <iostream>
#include <sstream>

```

Include dependency graph for ThreadPoolTest.cpp:



Functions

- int [main](#) ()

6.56.1 Function Documentation

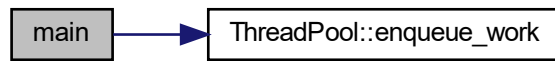
6.56.1.1 main()

```
int main (
    void )
```

Definition at line 9 of file [ThreadPoolTest.cpp](#).

```
00010 {
00011     srand(0);
00012     auto begin = chrono::high_resolution_clock::now();
00013
00014     auto numThreads = thread::hardware_concurrency();
00015
00016     static std::exception_ptr globalExceptionPtr = nullptr;
00017
00018     try
00019     {
00020         ThreadPool pool(numThreads);
00021
00022         cout << "queueing up some random work\n";
00023         for (size_t i = 0; i < 1000; i++)
00024         {
00025             // pass in lambda of some random work
00026             pool.enqueue_work([i]()
00027             {
00028                 try
00029                 {
00030                     size_t x;
00031                     size_t repetitions = 10 + (10 * (rand() % 5));
00032                     for (size_t j = 0; j < repetitions; j++)
00033                     {
00034                         x = i + (rand() % 200); // pretty good chance of generating a few
00035                                                 // exceptions every few runs
00036                     }
00037                     if (x > 1000)
00038                     {
00039                         stringstream id;
00040                         id << this_thread::get_id();
00041                         throw runtime_error(id.str().c_str());
00042                     }
00043                 } catch (...)
00044                 {
00045                     globalExceptionPtr = current_exception();
00046                 }
00047             });
00048             if (globalExceptionPtr)
00049             {
00050                 try
00051                 {
00052                     rethrow_exception(globalExceptionPtr);
00053                 } catch (const exception &e)
00054                 {
00055                     cout << "caught exception in job from thread: " << e.what() << "\n";
00056                 }
00057             }
00058         }
00059         auto end = chrono::high_resolution_clock::now();
00060         auto duration = chrono::duration_cast<chrono::milliseconds>(end - begin);
00061         cout << "pool duration = " << duration.count() / 1000.f << " seconds\n";
00062     } catch (const exception &ex)
00063     {
00064         cout << "caught exception from thread pool: " << ex.what() << "\n";
00065     }
00066
00067     return 0;
00068 }
```

Here is the call graph for this function:



6.57 ThreadPoolTest.cpp

[Go to the documentation of this file.](#)

```

00001 #include "ThreadPool.h"
00002
00003 #include <thread>
00004 #include <iostream>
00005 #include <sstream>
00006
00007 using namespace std;
00008
00009 int main()
00010 {
00011     srand(0);
00012     auto begin = chrono::high_resolution_clock::now();
00013
00014     auto numThreads = thread::hardware_concurrency();
00015
00016     static std::exception_ptr globalExceptionPtr = nullptr;
00017
00018     try
00019     {
00020         ThreadPool pool(numThreads);
00021
00022         cout << "queueing up some random work\n";
00023         for (size_t i = 0; i < 1000; i++)
00024         {
00025             // pass in lambda of some random work
00026             pool.enqueue_work([i]()
00027             {
00028                 try
00029                 {
00030                     size_t x;
00031                     size_t repetitions = 10 + (10 * (rand() % 5));
00032                     for (size_t j = 0; j < repetitions; j++)
00033                     {
00034                         x = i + (rand() % 200); // pretty good chance of generating a few
00035                                                 // exceptions every few runs
00036                     }
00037                     if (x > 1000)
00038                     {
00039                         stringstream id;
00040                         id << this_thread::get_id();
00041                         throw runtime_error(id.str().c_str());
00042                     }
00043                     catch (...)
00044                     {
00045                         globalExceptionPtr = current_exception();
00046                     }
00047                 }
00048             });
00049             if (globalExceptionPtr)
00050             {
00051                 try
00052                 {
00053                     rethrow_exception(globalExceptionPtr);
00054                 } catch (const exception &e)
00055                 {
00056                     cout << "caught exception in job from thread: " << e.what() << "\n";
00057                 }
00058             }
00059         }
00060         auto end = chrono::high_resolution_clock::now();
00061         auto duration = chrono::duration_cast<chrono::milliseconds>(end - begin);
00062         cout << "pool duration = " << duration.count() / 1000.f << " seconds\n";

```

```

00062     } catch (const exception &ex)
00063     {
00064         cout << "caught exception from thread pool: " << ex.what() << "\n";
00065     }
00066
00067     return 0;
00068 }

```

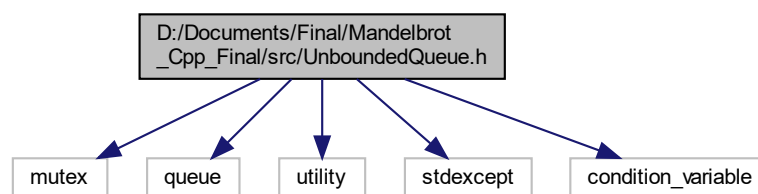
6.58 D:/Documents/Final/Mandelbrot_Cpp_Final/src/UnboundedQueue.h File Reference

```

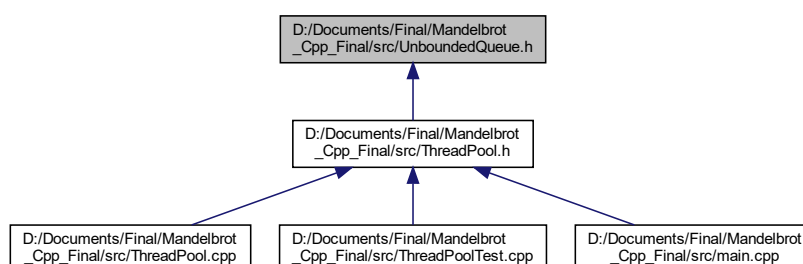
#include <mutex>
#include <queue>
#include <utility>
#include <stdexcept>
#include <condition_variable>

```

Include dependency graph for UnboundedQueue.h:



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



Classes

- class [UnboundedQueue< T >](#)

6.59 UnboundedQueue.h

[Go to the documentation of this file.](#)

```

00001 #ifndef C_____UNBOUNDEDQUEUE_H_
00002 #define C_____UNBOUNDEDQUEUE_H_
00003
00004 #include <mutex>
00005 #include <queue>
00006 #include <utility>
00007 #include <stdexcept>
00008 #include <condition_variable>
00009
00010 template<typename T>
00011 class UnboundedQueue
00012 {
00013 public:
00014
00015     explicit UnboundedQueue(bool block = true);
00016
00017     ~UnboundedQueue();
00018
00019     void push(const T& item);
00020
00021     void push(T&& item);
00022
00023     template<typename... ARGS>
00024     void emplace(ARGS&&... args);
00025
00026     bool try_push(const T& item);
00027
00028     bool try_push(T&& item);
00029
00030     bool pop(T& item);
00031
00032     bool try_pop(T& item);
00033
00034     std::size_t size() const;
00035
00036     bool empty() const;
00037
00038     void block();
00039
00040     void unblock();
00041
00042     bool blocking() const;
00043
00044 private:
00045
00046     using queue_t = std::queue<T>;
00047     queue_t queue;
00048
00049     bool is_block;
00050
00051     mutable std::mutex queueLock;
00052
00053     std::condition_variable condition;
00054 };
00055
00056 template<typename T>
00057 UnboundedQueue<T>::UnboundedQueue(bool block) : is_block(block)
00058 {}
00059
00060 template<typename T>
00061 void UnboundedQueue<T>::push(const T &item)
00062 {
00063     {
00064         std::scoped_lock guard(queueLock);
00065         queue.push(item);
00066     }
00067     condition.notify_one();
00068 }
00069
00070 template<typename T>
00071 void UnboundedQueue<T>::push(T &&item)
00072 {
00073     {
00074         std::scoped_lock guard(queueLock);
00075         queue.push(std::move(item));
00076     }
00077     condition.notify_one();
00078 }
00079
00080 template<typename T>

```

```

00101 template<typename... ARGS>
00102 void UnboundedQueue<T>::emplace(ARGS &&... args)
00103 {
00104     {
00105         std::scoped_lock guard(queueLock);
00106         queue.emplace(std::forward<>(args)...);
00107     }
00108     condition.notify_one();
00109 }
00110
00111 template<typename T>
00112 bool UnboundedQueue<T>::try_push(const T &item)
00113 {
00114     {
00115         std::unique_lock guard(queueLock, std::try_to_lock);
00116         if (!guard)
00117         {
00118             return false;
00119         }
00120         queue.push(item);
00121     }
00122     condition.notify_one();
00123     return true;
00124 }
00125
00126 template<typename T>
00127 bool UnboundedQueue<T>::try_push(T &&item)
00128 {
00129     {
00130         std::unique_lock guard(queueLock, std::try_to_lock);
00131         if (!guard)
00132         {
00133             return false;
00134         }
00135         queue.push(std::move(item));
00136     }
00137     condition.notify_one();
00138     return true;
00139 }
00140
00141 template<typename T>
00142 bool UnboundedQueue<T>::pop(T &item)
00143 {
00144     std::unique_lock guard(queueLock);
00145     condition.wait(guard, [&] () {return !queue.empty() || !is_block;});
00146     if (queue.empty())
00147     {
00148         return false;
00149     }
00150     item = std::move(queue.front());
00151     queue.pop();
00152     return true;
00153 }
00154
00155 template<typename T>
00156 bool UnboundedQueue<T>::try_pop(T &item)
00157 {
00158     std::unique_lock guard(queueLock, std::try_to_lock);
00159     if (!guard || queue.empty())
00160     {
00161         return false;
00162     }
00163     item = std::move(queue.front());
00164     queue.pop();
00165     return true;
00166 }
00167
00168 template<typename T>
00169 std::size_t UnboundedQueue<T>::size() const
00170 {
00171     std::scoped_lock guard(queueLock);
00172     return queue.size();
00173 }
00174
00175 template<typename T>
00176 bool UnboundedQueue<T>::empty() const
00177 {
00178     std::scoped_lock guard(queueLock);
00179     return queue.empty();
00180 }
00181
00182 template<typename T>
00183 void UnboundedQueue<T>::block()
00184 {
00185     std::scoped_lock guard(queueLock);
00186     is_block = true;
00187 }

```

```
00188
00189 template<typename T>
00190 void UnboundedQueue<T>::unlock()
00191 {
00192     {
00193         std::scoped_lock guard(queueLock);
00194         is_block = false;
00195     }
00196     condition.notify_all();
00197 }
00198
00199 template<typename T>
00200 bool UnboundedQueue<T>::blocking() const
00201 {
00202     std::scoped_lock guard(queueLock);
00203     return is_block;
00204 }
00205
00206 template<typename T>
00207 UnboundedQueue<T>::~UnboundedQueue()
00208 = default;
00209
00210
00211
00212 #endif //C_____UNBOUNDEDQUEUE_H_
```


Index

- `__has_include`
 - `CMakeCCompilerId.c`, [83](#)
 - `CMakeCXXCompilerId.cpp`, [97](#)
 - `~Colorization`
 - `Colorization`, [12](#)
 - `~InsideColor`
 - `InsideColor`, [17](#)
 - `~LineColor`
 - `LineColor`, [21](#)
 - `~Mandelbrot`
 - `Mandelbrot`, [26](#)
 - `~NormalMap`
 - `NormalMap`, [48](#)
 - `~Shading`
 - `Shading`, [62](#)
 - `~Striping`
 - `Striping`, [66](#)
 - `~ThreadPool`
 - `ThreadPool`, [71](#)
 - `~UnboundedQueue`
 - `UnboundedQueue< T >`, [76](#)
- `a`
 - `Mandelbrot`, [36](#)
- `angle`
 - `Neumorphic`, [45](#)
- `ARCHITECTURE_ID`
 - `CMakeCCompilerId.c`, [84](#)
 - `CMakeCXXCompilerId.cpp`, [97](#)
- `average`
 - `Mandelbrot`, [27](#)
 - `Striping`, [68](#)
- `block`
 - `UnboundedQueue< T >`, [76](#)
- `blocking`
 - `UnboundedQueue< T >`, [76](#)
- `bulb`
 - `Mandelbrot`, [36](#)
- `c`
 - `Mandelbrot`, [36](#)
- `C_DIALECT`
 - `CMakeCCompilerId.c`, [84](#)
- `calculate`
 - `Neumorphic`, [43](#)
 - `NormalMap`, [49](#)
- `calculate_b`
 - `InsideColor`, [17](#)
 - `LineColor`, [21](#)
 - `Shading`, [62](#)
 - `Striping`, [66](#)
- `calculate_bw`
 - `InsideColor`, [17](#)
 - `LineColor`, [21](#)
 - `Shading`, [62](#)
 - `Striping`, [66](#)
- `calculate_g`
 - `InsideColor`, [18](#)
 - `LineColor`, [22](#)
 - `Shading`, [62](#)
 - `Striping`, [67](#)
- `calculate_r`
 - `InsideColor`, [18](#)
 - `LineColor`, [22](#)
 - `Shading`, [63](#)
 - `Striping`, [67](#)
- `cardioid`
 - `Mandelbrot`, [36](#)
- `close`
 - `PPM`, [54](#)
- `CMakeCCompilerId.c`
 - `__has_include`, [83](#)
 - `ARCHITECTURE_ID`, [84](#)
 - `C_DIALECT`, [84](#)
 - `COMPILER_ID`, [84](#)
 - `DEC`, [84](#)
 - `HEX`, [84](#)
 - `info_arch`, [86](#)
 - `info_compiler`, [86](#)
 - `info_language_dialect_default`, [86](#)
 - `info_platform`, [86](#)
 - `main`, [85](#)
 - `PLATFORM_ID`, [85](#)
 - `STRINGIFY`, [85](#)
 - `STRINGIFY_HELPER`, [85](#)
- `CMakeCXXCompilerId.cpp`
 - `__has_include`, [97](#)
 - `ARCHITECTURE_ID`, [97](#)
 - `COMPILER_ID`, [97](#)
 - `CXX_STD`, [97](#)
 - `DEC`, [97](#)
 - `HEX`, [98](#)
 - `info_arch`, [99](#)
 - `info_compiler`, [99](#)
 - `info_language_dialect_default`, [100](#)
 - `info_platform`, [100](#)
 - `main`, [99](#)
 - `PLATFORM_ID`, [98](#)

- STRINGIFY, 98
- STRINGIFY_HELPER, 98
- Colorization, 11
 - ~Colorization, 12
 - Colorization, 13
 - get_max_color_value, 13
 - get_min_color_value, 13
 - get_type, 13
 - maxColorValue, 14
 - minColorValue, 14
 - type, 14
- colorize_bw
 - Mandelbrot, 27
- comment
 - PPM, 58
- COMPILER_ID
 - CMakeCCompilerId.c, 84
 - CMakeCXXCompilerId.cpp, 97
- condition
 - UnboundedQueue< T >, 80
- count
 - ThreadPool, 73
- countMult
 - ThreadPool, 73
- current_pixel
 - Mandelbrot, 28
- cxMax
 - Mandelbrot, 36
- cxMin
 - Mandelbrot, 36
- CXX_STD
 - CMakeCXXCompilerId.cpp, 97
- cyMax
 - Mandelbrot, 37
- cyMin
 - Mandelbrot, 37
- d
 - Mandelbrot, 37
- D:/Documents/Final/Mandelbrot_Cpp_Final/cmake-
 - build-debug/CMakeFiles/3.21.1/CompilerIdC/CMakeCCompilerId.c, 83, 87
- D:/Documents/Final/Mandelbrot_Cpp_Final/cmake-
 - build-debug/CMakeFiles/3.21.1/CompilerIdCXX/CMakeCXXCompilerId.cpp, 96, 100
- D:/Documents/Final/Mandelbrot_Cpp_Final/cmake-
 - build-debug/CMakeFiles/FindOpenMP/OpenMPCheckVersion.c, 109, 111
- D:/Documents/Final/Mandelbrot_Cpp_Final/cmake-
 - build-debug/CMakeFiles/FindOpenMP/OpenMPCheckVersion.cpp, 111, 112
- D:/Documents/Final/Mandelbrot_Cpp_Final/cmake-
 - build-debug/CMakeFiles/FindOpenMP/OpenMPTryFlag.c, 113
- D:/Documents/Final/Mandelbrot_Cpp_Final/cmake-
 - build-debug/CMakeFiles/FindOpenMP/OpenMPTryFlag.cpp, 114
- D:/Documents/Final/Mandelbrot_Cpp_Final/README.md, 115
- D:/Documents/Final/Mandelbrot_Cpp_Final/src/Colorization.cpp, 115
- D:/Documents/Final/Mandelbrot_Cpp_Final/src/Colorization.h, 116
- D:/Documents/Final/Mandelbrot_Cpp_Final/src/InsideColor.cpp, 117
- D:/Documents/Final/Mandelbrot_Cpp_Final/src/InsideColor.h, 118, 119
- D:/Documents/Final/Mandelbrot_Cpp_Final/src/LineColor.cpp, 120
- D:/Documents/Final/Mandelbrot_Cpp_Final/src/LineColor.h, 121, 122
- D:/Documents/Final/Mandelbrot_Cpp_Final/src/main.cpp, 122, 124
- D:/Documents/Final/Mandelbrot_Cpp_Final/src/Mandelbrot.cpp, 126, 127
- D:/Documents/Final/Mandelbrot_Cpp_Final/src/Mandelbrot.h, 130, 131
- D:/Documents/Final/Mandelbrot_Cpp_Final/src/Neumorphic.cpp, 133
- D:/Documents/Final/Mandelbrot_Cpp_Final/src/Neumorphic.h, 134, 135
- D:/Documents/Final/Mandelbrot_Cpp_Final/src/NormalMap.cpp, 136
- D:/Documents/Final/Mandelbrot_Cpp_Final/src/NormalMap.h, 136, 137
- D:/Documents/Final/Mandelbrot_Cpp_Final/src/PPM.cpp, 138
- D:/Documents/Final/Mandelbrot_Cpp_Final/src/PPM.h, 139, 140
- D:/Documents/Final/Mandelbrot_Cpp_Final/src/Shading.cpp, 141, 142
- D:/Documents/Final/Mandelbrot_Cpp_Final/src/Shading.h, 143
- D:/Documents/Final/Mandelbrot_Cpp_Final/src/Striping.cpp, 144, 145
- D:/Documents/Final/Mandelbrot_Cpp_Final/src/Striping.h, 145, 146
- D:/Documents/Final/Mandelbrot_Cpp_Final/src/ThreadPool.cpp, 147
- D:/Documents/Final/Mandelbrot_Cpp_Final/src/ThreadPool.h, 148, 149
- D:/Documents/Final/Mandelbrot_Cpp_Final/src/ThreadPoolTest.cpp, 150, 151
- D:/Documents/Final/Mandelbrot_Cpp_Final/src/UnboundedQueue.h, 153, 154
- do
 - Mandelbrot, 37
 - Neumorphic, 45
 - Striping, 45
 - Mandelbrot, 37
- DEC
 - CMakeCCompilerId.c, 84
 - CMakeCXXCompilerId.cpp, 97
- describe_border
 - Mandelbrot, 29
- dot_product
 - NormalMap, 49

- emplace
 - UnboundedQueue< T >, 77
- empty
 - UnboundedQueue< T >, 77
- enqueue_task
 - ThreadPool, 72
- enqueue_work
 - ThreadPool, 72
- escapeRadius
 - Mandelbrot, 37
- fileName
 - PPM, 58
- get_angle
 - Neumorphic, 44
- get_average
 - Striping, 67
- get_c
 - Mandelbrot, 29
- get_heightFactor
 - Neumorphic, 44
- get_max_color_value
 - Colorization, 13
 - Shading, 63
- get_max_val
 - NormalMap, 50
- get_min_color_value
 - Colorization, 13
 - Shading, 63
- get_min_val
 - NormalMap, 50
- get_reflection
 - Neumorphic, 44
- get_t
 - Mandelbrot, 30
- get_type
 - Colorization, 13
 - NormalMap, 50
- header
 - PPM, 58
- height
 - Mandelbrot, 38
 - PPM, 59
- heightFactor
 - Neumorphic, 45
- HEX
 - CMakeCCompilerId.c, 84
 - CMakeCXXCompilerId.cpp, 98
- image
 - PPM, 59
- in_border
 - Mandelbrot, 30
- in_set
 - Mandelbrot, 31
- index
 - ThreadPool, 73
- info_arch
 - CMakeCCompilerId.c, 86
 - CMakeCXXCompilerId.cpp, 99
- info_compiler
 - CMakeCCompilerId.c, 86
 - CMakeCXXCompilerId.cpp, 99
- info_language_dialect_default
 - CMakeCCompilerId.c, 86
 - CMakeCXXCompilerId.cpp, 100
- info_platform
 - CMakeCCompilerId.c, 86
 - CMakeCXXCompilerId.cpp, 100
- init_stream
 - PPM, 55
- InsideColor, 15
 - ~InsideColor, 17
 - calculate_b, 17
 - calculate_bw, 17
 - calculate_g, 18
 - calculate_r, 18
 - InsideColor, 17
- interpolate
 - Mandelbrot, 31
- is_block
 - UnboundedQueue< T >, 80
- iSkip
 - Mandelbrot, 38
- iter
 - Mandelbrot, 38
- iterate
 - Mandelbrot, 32
- iterMax
 - Mandelbrot, 38
- LineColor, 19
 - ~LineColor, 21
 - calculate_b, 21
 - calculate_bw, 21
 - calculate_g, 22
 - calculate_r, 22
 - LineColor, 21
- M_PI
 - Mandelbrot.h, 131
- magic
 - PPM, 59
- main
 - CMakeCCompilerId.c, 85
 - CMakeCXXCompilerId.cpp, 99
 - main.cpp, 123
 - OpenMPCheckVersion.c, 110
 - OpenMPCheckVersion.cpp, 112
 - OpenMPTryFlag.c, 113
 - OpenMPTryFlag.cpp, 114
 - ThreadPoolTest.cpp, 151
- main.cpp
 - main, 123
- Mandelbrot, 23
 - ~Mandelbrot, 26

- a, 36
- average, 27
- bulb, 36
- c, 36
- cardioid, 36
- colorize_bw, 27
- current_pixel, 28
- cxMax, 36
- cxMin, 36
- cyMax, 37
- cyMin, 37
- d, 37
- dC, 37
- de, 37
- describe_border, 29
- escapeRadius, 37
- get_c, 29
- get_t, 30
- height, 38
- in_border, 30
- in_set, 31
- interpolate, 31
- iSkip, 38
- iter, 38
- iterate, 32
- iterMax, 38
- Mandelbrot, 25, 26
- operator<<, 35
- pixHeight, 38
- pixWidth, 38
- prevA, 39
- pX, 39
- pY, 39
- q, 39
- r, 39
- reset, 33
- set_border, 33
- set_image, 34
- set_iSkip, 34
- set_plane, 34
- set_stripe_density, 34
- shade, 39
- shape_check, 35
- stripeDensity, 40
- thin, 40
- width, 40
- z, 40
- Mandelbrot.cpp
 - operator<<, 126
- Mandelbrot.h
 - M_PI, 131
- maxColorValue
 - Colorization, 14
- maxMapVal
 - NormalMap, 50
- minColorValue
 - Colorization, 14
- minMapVal
 - NormalMap, 50
- Neumorphic, 41
 - angle, 45
 - calculate, 43
 - dC, 45
 - get_angle, 44
 - get_heightFactor, 44
 - get_reflection, 44
 - heightFactor, 45
 - Neumorphic, 43
 - reflection, 45
 - u, 45
 - v, 46
 - z, 46
- NormalMap, 46
 - ~NormalMap, 48
 - calculate, 49
 - dot_product, 49
 - get_max_val, 50
 - get_min_val, 50
 - get_type, 50
 - maxMapVal, 50
 - minMapVal, 50
 - NormalMap, 49
 - type, 51
- ompver_str
 - OpenMPCheckVersion.c, 110
 - OpenMPCheckVersion.cpp, 112
- OpenMPCheckVersion.c
 - main, 110
 - ompver_str, 110
- OpenMPCheckVersion.cpp
 - main, 112
 - ompver_str, 112
- OpenMPTryFlag.c
 - main, 113
- OpenMPTryFlag.cpp
 - main, 114
- operator<<
 - Mandelbrot, 35
 - Mandelbrot.cpp, 126
- operator=
 - PPM, 55
- outputDirectory
 - PPM, 59
- pixHeight
 - Mandelbrot, 38
- pixMaxVal
 - PPM, 59
- pixWidth
 - Mandelbrot, 38
- PLATFORM_ID
 - CMakeCCompilerId.c, 85
 - CMakeCXXCompilerId.cpp, 98
- pop
 - UnboundedQueue< T >, 77

- PPM, 51
 - close, 54
 - comment, 58
 - fileName, 58
 - header, 58
 - height, 59
 - image, 59
 - init_stream, 55
 - magic, 59
 - operator=, 55
 - outputDirectory, 59
 - pixMaxVal, 59
 - PPM, 53, 54
 - set_comment, 56
 - set_height, 56
 - set_outputDirectory, 56
 - set_width, 57
 - subPixel, 59
 - width, 60
 - write_header, 57
 - write_row, 57
- prevA
 - Mandelbrot, 39
- process
 - ThreadPool, 70
- push
 - UnboundedQueue< T >, 78
- pX
 - Mandelbrot, 39
- pY
 - Mandelbrot, 39
- q
 - Mandelbrot, 39
- queue
 - ThreadPool, 70
 - UnboundedQueue< T >, 81
- queue_t
 - UnboundedQueue< T >, 75
- queueLock
 - UnboundedQueue< T >, 81
- queues
 - ThreadPool, 73
- queueVec
 - ThreadPool, 70
- r
 - Mandelbrot, 39
- reflection
 - Neumorphic, 45
 - Striping, 68
- reset
 - Mandelbrot, 33
- set_border
 - Mandelbrot, 33
- set_comment
 - PPM, 56
- set_height
 - PPM, 56
- set_image
 - Mandelbrot, 34
- set_iSkip
 - Mandelbrot, 34
- set_outputDirectory
 - PPM, 56
- set_plane
 - Mandelbrot, 34
- set_stripe_density
 - Mandelbrot, 34
- set_width
 - PPM, 57
- shade
 - Mandelbrot, 39
- Shading, 60
 - ~Shading, 62
 - calculate_b, 62
 - calculate_bw, 62
 - calculate_g, 62
 - calculate_r, 63
 - get_max_color_value, 63
 - get_min_color_value, 63
 - Shading, 62
 - type, 63
- shape_check
 - Mandelbrot, 35
- size
 - UnboundedQueue< T >, 79
- STRINGIFY
 - CMakeCCompilerId.c, 85
 - CMakeCXXCompilerId.cpp, 98
- STRINGIFY_HELPER
 - CMakeCCompilerId.c, 85
 - CMakeCXXCompilerId.cpp, 98
- stripeDensity
 - Mandelbrot, 40
- Striping, 64
 - ~Striping, 66
 - average, 68
 - calculate_b, 66
 - calculate_bw, 66
 - calculate_g, 67
 - calculate_r, 67
 - get_average, 67
 - reflection, 68
 - Striping, 66
- subPixel
 - PPM, 59
- thin
 - Mandelbrot, 40
- ThreadPool, 68
 - ~ThreadPool, 71
 - count, 73
 - countMult, 73
 - enqueue_task, 72
 - enqueue_work, 72
 - index, 73

- process, [70](#)
- queue, [70](#)
- queues, [73](#)
- queueVec, [70](#)
- ThreadPool, [70](#)
- Threads, [70](#)
- threads, [73](#)
- ThreadPoolTest.cpp
 - main, [151](#)
- Threads
 - ThreadPool, [70](#)
- threads
 - ThreadPool, [73](#)
- try_pop
 - UnboundedQueue< T >, [79](#)
- try_push
 - UnboundedQueue< T >, [79](#)
- type
 - Colorization, [14](#)
 - NormalMap, [51](#)
 - Shading, [63](#)
- u
 - Neumorphic, [45](#)
- unblock
 - UnboundedQueue< T >, [80](#)
- UnboundedQueue
 - UnboundedQueue< T >, [76](#)
- UnboundedQueue< T >, [74](#)
 - ~UnboundedQueue, [76](#)
 - block, [76](#)
 - blocking, [76](#)
 - condition, [80](#)
 - emplace, [77](#)
 - empty, [77](#)
 - is_block, [80](#)
 - pop, [77](#)
 - push, [78](#)
 - queue, [81](#)
 - queue_t, [75](#)
 - queueLock, [81](#)
 - size, [79](#)
 - try_pop, [79](#)
 - try_push, [79](#)
 - unblock, [80](#)
 - UnboundedQueue, [76](#)
- v
 - Neumorphic, [46](#)
- width
 - Mandelbrot, [40](#)
 - PPM, [60](#)
- write_header
 - PPM, [57](#)
- write_row
 - PPM, [57](#)
- z