

ShapeLib

Generated by Doxygen 1.9.3

1 AK5PC - Shape 2D	1
2 LICENSE	3
3 Namespace Index	5
3.1 Namespace List	5
4 Hierarchical Index	7
4.1 Class Hierarchy	7
5 Class Index	9
5.1 Class List	9
6 File Index	11
6.1 File List	11
7 Namespace Documentation	13
7.1 ShapeLib Namespace Reference	13
7.1.1 Function Documentation	13
7.1.1.1 operator<<()	13
8 Class Documentation	15
8.1 Point Class Reference	15
8.1.1 Detailed Description	15
8.1.2 Constructor & Destructor Documentation	16
8.1.2.1 Point()	16
8.1.3 Member Function Documentation	16
8.1.3.1 horizontalDistance()	16
8.1.3.2 verticalDistance()	17
8.1.4 Member Data Documentation	17
8.1.4.1 x	17
8.1.4.2 y	17
8.2 ShapeLib::Rectangle Class Reference	18
8.2.1 Detailed Description	19
8.2.2 Constructor & Destructor Documentation	19
8.2.2.1 Rectangle() [1/2]	19
8.2.2.2 Rectangle() [2/2]	19
8.2.3 Member Function Documentation	20
8.2.3.1 getHeight()	20
8.2.3.2 getP1()	20
8.2.3.3 getP2()	21
8.2.3.4 getWidth()	21
8.2.3.5 printInfo()	21
8.2.3.6 recalculateDim()	21
8.2.3.7 setP1()	21

8.2.3.8 setP2()	22
8.2.4 Member Data Documentation	22
8.2.4.1 dimValid	22
8.2.4.2 height	22
8.2.4.3 p1	23
8.2.4.4 p2	23
8.2.4.5 width	23
8.3 ShapeLib::Shape Class Reference	23
8.3.1 Detailed Description	24
8.3.2 Constructor & Destructor Documentation	24
8.3.2.1 Shape()	24
8.3.3 Member Function Documentation	24
8.3.3.1 getID()	24
8.3.3.2 printInfo()	25
8.3.4 Member Data Documentation	25
8.3.4.1 ID	25
9 File Documentation	27
9.1 CMakeCCompilerId.c File Reference	27
9.1.1 Macro Definition Documentation	27
9.1.1.1 __has_include	28
9.1.1.2 ARCHITECTURE_ID	28
9.1.1.3 C_VERSION	28
9.1.1.4 COMPILER_ID	28
9.1.1.5 DEC	28
9.1.1.6 HEX	28
9.1.1.7 PLATFORM_ID	29
9.1.1.8 STRINGIFY	29
9.1.1.9 STRINGIFY_HELPER	29
9.1.2 Function Documentation	29
9.1.2.1 main()	29
9.1.3 Variable Documentation	29
9.1.3.1 info_arch	29
9.1.3.2 info_compiler	29
9.1.3.3 info_language_extensions_default	30
9.1.3.4 info_language_standard_default	30
9.1.3.5 info_platform	30
9.2 CMakeCXXCompilerId.cpp File Reference	30
9.2.1 Macro Definition Documentation	31
9.2.1.1 __has_include	31
9.2.1.2 ARCHITECTURE_ID	31
9.2.1.3 COMPILER_ID	31

9.2.1.4 CXX_STD	31
9.2.1.5 DEC	31
9.2.1.6 HEX	32
9.2.1.7 PLATFORM_ID	32
9.2.1.8 STRINGIFY	32
9.2.1.9 STRINGIFY_HELPER	32
9.2.2 Function Documentation	32
9.2.2.1 main()	32
9.2.3 Variable Documentation	32
9.2.3.1 info_arch	33
9.2.3.2 info_compiler	33
9.2.3.3 info_language_extensions_default	33
9.2.3.4 info_language_standard_default	33
9.2.3.5 info_platform	33
9.3 LICENSE.MD File Reference	33
9.4 main.cpp File Reference	33
9.4.1 Function Documentation	34
9.4.1.1 main()	34
9.5 README.md File Reference	34
9.6 point.h File Reference	34
9.7 point.h	34
9.8 rectangle.h File Reference	34
9.9 rectangle.h	35
9.10 shape.h File Reference	36
9.11 shape.h	36
9.12 point.cpp File Reference	36
9.13 rectangle.cpp File Reference	36
9.14 shape.cpp File Reference	37
Index	39

Chapter 1

AK5PC - Shape 2D

Demonstration project for AK5PC course.

Chapter 2

LICENSE

MIT License

Copyright (c) 2022 Ing. Peter Janku, Ph.D.

Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

Chapter 3

Namespace Index

3.1 Namespace List

Here is a list of all namespaces with brief descriptions:

ShapeLib	13
--------------------------	-------	----

Chapter 4

Hierarchical Index

4.1 Class Hierarchy

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

Point	15
ShapeLib::Shape	23
ShapeLib::Rectangle	18

Chapter 5

Class Index

5.1 Class List

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

Point	Simple class representing one point in cartesian coordinates	15
ShapeLib::Rectangle	Simple class representing a rectangle defined by two points	18
ShapeLib::Shape	Interface for any 2D shape in this library	23

Chapter 6

File Index

6.1 File List

Here is a list of all files with brief descriptions:

CMakeCCompilerId.c	27
CMakeCXXCompilerId.cpp	30
main.cpp	33
point.h	34
rectangle.h	34
shape.h	36
point.cpp	36
rectangle.cpp	36
shape.cpp	37

Chapter 7

Namespace Documentation

7.1 ShapeLib Namespace Reference

Classes

- class [Rectangle](#)
Simple class representing a rectangle defined by two points.
- class [Shape](#)
Interface for any 2D shape in this library.

Functions

- `std::ostream & operator<< (std::ostream &stream, Rectangle &rect)`

7.1.1 Function Documentation

7.1.1.1 `operator<<()`

```
std::ostream & ShapeLib::operator<< (  
    std::ostream & stream,  
    ShapeLib::Rectangle & rect )
```


Chapter 8

Class Documentation

8.1 Point Class Reference

Simple class representing one point in cartesian coordinates.

```
#include <point.h>
```

Public Member Functions

- [Point](#) (int [x](#), int [y](#))
First parametric constructor.

Static Public Member Functions

- static int [horizontalDistance](#) (const [Point](#) &p1, const [Point](#) &p2)
Static function for horizontal distance of two point estimation.
- static int [verticalDistance](#) (const [Point](#) &p1, const [Point](#) &p2)
Static function for vertical distance of two point estimation.

Public Attributes

- int [x](#)
coordinate X
- int [y](#)
coordinate Y

8.1.1 Detailed Description

Simple class representing one point in cartesian coordinates.

8.1.2 Constructor & Destructor Documentation

8.1.2.1 Point()

```
Point::Point (
    int x,
    int y ) [inline]
```

First parametric constructor.

Because at least one parametric constructor is created, the compiler doesn't create the default constructor.

Parameters

<i>x</i>	coordinate
<i>y</i>	coordinate

8.1.3 Member Function Documentation

8.1.3.1 horizontalDistance()

```
int Point::horizontalDistance (
    const Point & p1,
    const Point & p2 ) [static]
```

Static function for horizontal distance of two point estimation.

See also

[Point](#)

Parameters

<i>p1</i>	first input point
<i>p2</i>	second input point

Returns

horizontal distance between provided pints

8.1.3.2 verticalDistance()

```
int Point::verticalDistance (
    const Point & p1,
    const Point & p2 ) [static]
```

Static function for vertical distance of two point estimation.

See also

[Point](#)

Parameters

<i>p1</i>	first input point
<i>p2</i>	second input point

Returns

vertical distance between provided points

8.1.4 Member Data Documentation

8.1.4.1 x

```
int Point::x
```

coordinate X

8.1.4.2 y

```
int Point::y
```

coordinate Y

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

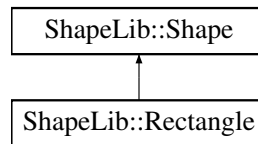
- [point.h](#)
- [point.cpp](#)

8.2 ShapeLib::Rectangle Class Reference

Simple class representing a rectangle defined by two points.

```
#include <rectangle.h>
```

Inheritance diagram for ShapeLib::Rectangle:



Public Member Functions

- [Rectangle](#) (int id, int x1, int y1, int x2, int y2)
The rectangle constructor.
- [Rectangle](#) (int id, const [Point](#) &p1, const [Point](#) &p2)
The main rectangle constructor.
- void [printInfo](#) () const override
Virtual function for printing information about the shape into the console.
- int [getWidth](#) ()
This function returns the current width of the rectangle. If the already calculated width is incorrect(the rectangle points were redefined), the dimensions are recalculated at first.
- int [getHeight](#) ()
This function returns the current height of the rectangle. If the already calculated height is incorrect(the rectangle points were redefined), the dimensions are recalculated at first.
- const [Point](#) & [getP1](#) () const
Return the first point in 2D space, which defines the rectangle.
- const [Point](#) & [getP2](#) () const
Return the second point in 2D space, which defines the rectangle.
- void [setP1](#) (const [Point](#) &p1)
Setter for the first point of the rectangle.
- void [setP2](#) (const [Point](#) &p2)
Setter for the second point of the rectangle.

Protected Member Functions

- void [recalculateDim](#) ()
Recalculates the rectangle width and height.

Protected Attributes

- [Point](#) p1
- [Point](#) p2
The first point in 2D space.
- int [width](#)
Temporary storage of rectangle width. Don't have to be correct!!!!
- int [height](#)
Temporary storage of rectangle height. Don't have to be correct!!!!
- bool [dimValid](#) = false
Information if width and height is correct or not.

8.2.1 Detailed Description

Simple class representing a rectangle defined by two points.

See also

[Point](#)

8.2.2 Constructor & Destructor Documentation

8.2.2.1 Rectangle() [1/2]

```
ShapeLib::Rectangle::Rectangle (  
    int id,  
    int x1,  
    int y1,  
    int x2,  
    int y2 ) [inline]
```

The rectangle constructor.

This constructor is calling the second one with modified parameters. This can be done since C++11 standard and it's called constructor delegation.

Parameters

<i>id</i>	Rectangle ID
<i>x1</i>	X coordinate of first rectangle's point
<i>y1</i>	Y coordinate of first rectangle's point
<i>x2</i>	X coordinate of second rectangle's point
<i>y2</i>	Y coordinate of second rectangle's point

8.2.2.2 Rectangle() [2/2]

```
ShapeLib::Rectangle::Rectangle (  
    int id,  
    const Point & p1,  
    const Point & p2 ) [inline]
```

The main rectangle constructor.

This constructor stores all [Rectangle](#)'s internal members as well as members defined in [Shape](#) class. It also calls constructor of [Shape](#) class. Finally, it recalculates the value dimensions of the rectangle.

See also

[recalculateDim\(\)](#)

Parameters

<i>id</i>	
<i>p1</i>	
<i>p2</i>	

8.2.3 Member Function Documentation

8.2.3.1 getHeight()

```
int ShapeLib::Rectangle::getHeight ( )
```

This function returns the current height of the rectangle. If the already calculated height is incorrect(the rectangle points were redefined), the dimensions are recalculated at first.

See also

[recalculateDim\(\)](#)

Returns

current height of the rectangle.

8.2.3.2 getP1()

```
const Point & ShapeLib::Rectangle::getP1 ( ) const [inline]
```

Return the first point in 2D space, which defines the rectangle.

Returns

Const reference to the point

See also

[Point](#)

8.2.3.3 getP2()

```
const Point & ShapeLib::Rectangle::getP2 ( ) const [inline]
```

Return the second point in 2D space, which defines the rectangle.

Returns

Const reference to the point

See also

[Point](#)

8.2.3.4 getWidth()

```
int ShapeLib::Rectangle::getWidth ( )
```

This function returns the current width of the rectangle. If the already calculated width is incorrect(the rectangle points were redefined), the dimensions are recalculated at first.

See also

[recalculateDim\(\)](#)

Returns

current width of the rectangle.

8.2.3.5 printInfo()

```
void ShapeLib::Rectangle::printInfo ( ) const [override], [virtual]
```

Virtual function for printing information about the shape into the console.

Implements [ShapeLib::Shape](#).

8.2.3.6 recalculateDim()

```
void ShapeLib::Rectangle::recalculateDim ( ) [protected]
```

Recalculates the rectangle width and height.

Based on p1 and p2 this function recalculates the rectangle width and height and store it in internal temporary variables.

See also

[width](#)

[height](#)

8.2.3.7 setP1()

```
void ShapeLib::Rectangle::setP1 (
    const Point & p1 )
```

Setter for the first point of the rectangle.

Parameters

<i>p1</i>	point defining the rectangle.
-----------	-------------------------------

8.2.3.8 setP2()

```
void ShapeLib::Rectangle::setP2 (
    const Point & p2 )
```

Setter for the second point of the rectangle.

Parameters

<i>p2</i>	point defining the rectangle.
-----------	-------------------------------

8.2.4 Member Data Documentation**8.2.4.1 dimValid**

```
bool ShapeLib::Rectangle::dimValid = false [mutable], [protected]
```

Information if width and height is correct or not.

See also

[recalculateDim\(\)](#)

8.2.4.2 height

```
int ShapeLib::Rectangle::height [mutable], [protected]
```

Temporary storage of rectangle height. Don't have to be correct!!!!

See also

[dimValid](#)

8.2.4.3 p1

`Point ShapeLib::Rectangle::p1` [protected]

8.2.4.4 p2

`Point ShapeLib::Rectangle::p2` [protected]

The first point in 2D space.

<

The second point in 2D space

8.2.4.5 width

`int ShapeLib::Rectangle::width` [mutable], [protected]

Temporary storage of rectangle width. Don't have to be correct!!!!

See also

[dimValid](#)

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

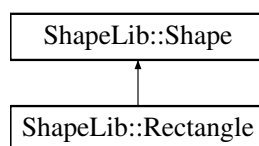
- [rectangle.h](#)
- [rectangle.cpp](#)

8.3 ShapeLib::Shape Class Reference

Interface for any 2D shape in this library.

```
#include <shape.h>
```

Inheritance diagram for ShapeLib::Shape:



Public Member Functions

- [Shape](#) (int id)
- virtual void [printInfo](#) () const =0
Virtual function for printing information about the shape into the console.
- int [getID](#) () const
Getter for shape's ID.

Protected Attributes

- int [ID](#)
internal ID of each shape

8.3.1 Detailed Description

Interface for any 2D shape in this library.

This function implements the basic ID management as well as virtual function [printInfo\(\)](#).

8.3.2 Constructor & Destructor Documentation

8.3.2.1 Shape()

```
ShapeLib::Shape::Shape (  
    int id ) [inline], [explicit]
```

@brief Constructor of [Shape](#) class

Parameters

<i>id</i>	an ID of the shape
-----------	--------------------

8.3.3 Member Function Documentation

8.3.3.1 getID()

```
int ShapeLib::Shape::getID ( ) const [inline]
```

Getter for shape's ID.

Returns

the ID of the shape

8.3.3.2 printInfo()

```
virtual void ShapeLib::Shape::printInfo ( ) const [pure virtual]
```

Virtual function for printing information about the shape into the console.

Implemented in [ShapeLib::Rectangle](#).

8.3.4 Member Data Documentation

8.3.4.1 ID

```
int ShapeLib::Shape::ID [protected]
```

internal ID of each shape

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

- [shape.h](#)

Chapter 9

File Documentation

9.1 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_VERSION`

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_standard_default`
- `const char * info_language_extensions_default`

9.1.1 Macro Definition Documentation

9.1.1.1 __has_include

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

9.1.1.2 ARCHITECTURE_ID

```
#define ARCHITECTURE_ID
```

9.1.1.3 C_VERSION

```
#define C_VERSION
```

9.1.1.4 COMPILER_ID

```
#define COMPILER_ID ""
```

9.1.1.5 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))
```

9.1.1.6 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))
```

9.1.1.7 PLATFORM_ID

```
#define PLATFORM_ID
```

9.1.1.8 STRINGIFY

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

9.1.1.9 STRINGIFY_HELPER

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

9.1.2 Function Documentation

9.1.2.1 main()

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

9.1.3 Variable Documentation

9.1.3.1 info_arch

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

9.1.3.2 info_compiler

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

9.1.3.3 info_language_extensions_default

```
const char* info_language_extensions_default
```

Initial value:

```
= "INFO" ":" "extensions_default["  
  "OFF"  
"]"
```

9.1.3.4 info_language_standard_default

```
const char* info_language_standard_default
```

Initial value:

```
=  
  "INFO" ":" "standard_default[" C_VERSION "]"
```

9.1.3.5 info_platform

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

9.2 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_standard_default`
- `const char * info_language_extensions_default`

9.2.1 Macro Definition Documentation

9.2.1.1 `__has_include`

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

9.2.1.2 `ARCHITECTURE_ID`

```
#define ARCHITECTURE_ID
```

9.2.1.3 `COMPILER_ID`

```
#define COMPILER_ID ""
```

9.2.1.4 `CXX_STD`

```
#define CXX_STD __cplusplus
```

9.2.1.5 `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))
```

9.2.1.6 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))
```

9.2.1.7 PLATFORM_ID

```
#define PLATFORM_ID
```

9.2.1.8 STRINGIFY

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

9.2.1.9 STRINGIFY_HELPER

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

9.2.2 Function Documentation

9.2.2.1 main()

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

9.2.3 Variable Documentation

9.2.3.1 info_arch

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

9.2.3.2 info_compiler

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

9.2.3.3 info_language_extensions_default

```
const char* info_language_extensions_default
```

Initial value:

```
= "INFO" ":" "extensions_default["  
  "OFF"  
"]"
```

9.2.3.4 info_language_standard_default

```
const char* info_language_standard_default
```

Initial value:

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

9.2.3.5 info_platform

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

9.3 LICENSE.MD File Reference

9.4 main.cpp File Reference

```
#include <iostream>  
#include "point.h"  
#include "rectangle.h"
```

Functions

- int [main](#) ()

9.4.1 Function Documentation

9.4.1.1 main()

```
int main ( )
```

9.5 README.md File Reference

9.6 point.h File Reference

Classes

- class [Point](#)
Simple class representing one point in cartesian coordinates.

9.7 point.h

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

```
1 //
2 // Created by Peter Janků on 08.10.2022.
3 //
4
5 #ifndef SHAPE2D_POINT_H
6 #define SHAPE2D_POINT_H
7
11 class Point {
12 public:
13
21     Point(int x, int y) : x(x), y(y) {}
22
23     static int horizontalDistance(const Point &p1, const Point &p2);
24
25     static int verticalDistance(const Point &p1, const Point &p2);
26
27     int x;
28     int y;
29 };
30
31
32 #endif //SHAPE2D_POINT_H
```

9.8 rectangle.h File Reference

```
#include <iostream>
#include "shape.h"
#include "point.h"
```


Classes

- class [ShapeLib::Rectangle](#)
Simple class representing a rectangle defined by two points.

Namespaces

- namespace [ShapeLib](#)

Functions

- `std::ostream & ShapeLib::operator<< (std::ostream &stream, Rectangle &rect)`

9.9 rectangle.h

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

```

1 //
2 // Created by Peter Janků on 08.10.2022.
3 //
4
5 #ifndef SHAPE2D_RECTANGLE_H
6 #define SHAPE2D_RECTANGLE_H
7
8 #include <iostream>
9 #include "shape.h"
10 #include "point.h"
11
12 namespace ShapeLib {
13     class Rectangle : public Shape {
14     public:
15
16         Rectangle(int id, int x1, int y1, int x2, int y2) : Rectangle(id, Point(x1, y1), Point(x2, y2))
17         {}
18
19         Rectangle(int id, const Point &p1, const Point &p2) : Shape(id), p1(p1), p2(p2) {
20             recalculateDim(); }
21
22         void printInfo() const override;
23
24         int getWidth();
25
26         int getHeight();
27
28         const Point &getP1() const { return p1; }
29
30         const Point &getP2() const { return p2; }
31
32     protected:
33
34         void recalculateDim();
35
36         Point p1;
37     public:
38         void setP1(const Point &p1);
39
40         void setP2(const Point &p2);
41
42     protected:
43         Point p2;
44         mutable int width;
45         mutable int height;
46         mutable bool dimValid = false;
47     };
48
49     std::ostream &operator<<(std::ostream &stream, Rectangle &rect);
50 }
51
52 #endif //SHAPE2D_RECTANGLE_H

```

9.10 shape.h File Reference

```
#include "point.h"
```

Classes

- class [ShapeLib::Shape](#)
Interface for any 2D shape in this library.

Namespaces

- namespace [ShapeLib](#)

9.11 shape.h

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

```
1 //
2 // Created by Peter Janků on 08.10.2022.
3 //
4
5 #ifndef SHAPE2D_SHAPE_H
6 #define SHAPE2D_SHAPE_H
7
8 #include "point.h"
9
10 namespace ShapeLib {
11     class Shape {
12     public:
13
14         explicit Shape(int id) : ID(id) {}
15
16         virtual void printInfo() const = 0;
17
18         int getID() const { return ID; }
19
20     protected:
21         int ID;
22     };
23 } // ShapeLib
24
25 #endif //SHAPE2D_SHAPE_H
```

9.12 point.cpp File Reference

```
#include "../include/point.h"
#include <cmath>
```

9.13 rectangle.cpp File Reference

```
#include <iostream>
#include "rectangle.h"
```

9.14 shape.cpp File Reference

```
#include "shape.h"
```

Namespaces

- namespace [ShapeLib](#)

Index

- [__has_include](#)
 - [CMakeCCompilerId.c, 27](#)
 - [CMakeCXXCompilerId.cpp, 31](#)
- [ARCHITECTURE_ID](#)
 - [CMakeCCompilerId.c, 28](#)
 - [CMakeCXXCompilerId.cpp, 31](#)
- [C_VERSION](#)
 - [CMakeCCompilerId.c, 28](#)
- [CMakeCCompilerId.c, 27](#)
 - [__has_include, 27](#)
 - [ARCHITECTURE_ID, 28](#)
 - [C_VERSION, 28](#)
 - [COMPILER_ID, 28](#)
 - [DEC, 28](#)
 - [HEX, 28](#)
 - [info_arch, 29](#)
 - [info_compiler, 29](#)
 - [info_language_extensions_default, 29](#)
 - [info_language_standard_default, 30](#)
 - [info_platform, 30](#)
 - [main, 29](#)
 - [PLATFORM_ID, 28](#)
 - [STRINGIFY, 29](#)
 - [STRINGIFY_HELPER, 29](#)
- [CMakeCXXCompilerId.cpp, 30](#)
 - [__has_include, 31](#)
 - [ARCHITECTURE_ID, 31](#)
 - [COMPILER_ID, 31](#)
 - [CXX_STD, 31](#)
 - [DEC, 31](#)
 - [HEX, 31](#)
 - [info_arch, 32](#)
 - [info_compiler, 33](#)
 - [info_language_extensions_default, 33](#)
 - [info_language_standard_default, 33](#)
 - [info_platform, 33](#)
 - [main, 32](#)
 - [PLATFORM_ID, 32](#)
 - [STRINGIFY, 32](#)
 - [STRINGIFY_HELPER, 32](#)
- [COMPILER_ID](#)
 - [CMakeCCompilerId.c, 28](#)
 - [CMakeCXXCompilerId.cpp, 31](#)
- [CXX_STD](#)
 - [CMakeCXXCompilerId.cpp, 31](#)
- [DEC](#)
 - [CMakeCCompilerId.c, 28](#)
- [CMakeCXXCompilerId.cpp, 31](#)
- [dimValid](#)
 - [ShapeLib::Rectangle, 22](#)
- [getHeight](#)
 - [ShapeLib::Rectangle, 20](#)
- [getID](#)
 - [ShapeLib::Shape, 24](#)
- [getP1](#)
 - [ShapeLib::Rectangle, 20](#)
- [getP2](#)
 - [ShapeLib::Rectangle, 20](#)
- [getWidth](#)
 - [ShapeLib::Rectangle, 21](#)
- [height](#)
 - [ShapeLib::Rectangle, 22](#)
- [HEX](#)
 - [CMakeCCompilerId.c, 28](#)
 - [CMakeCXXCompilerId.cpp, 31](#)
- [horizontalDistance](#)
 - [Point, 16](#)
- [ID](#)
 - [ShapeLib::Shape, 25](#)
- [info_arch](#)
 - [CMakeCCompilerId.c, 29](#)
 - [CMakeCXXCompilerId.cpp, 32](#)
- [info_compiler](#)
 - [CMakeCCompilerId.c, 29](#)
 - [CMakeCXXCompilerId.cpp, 33](#)
- [info_language_extensions_default](#)
 - [CMakeCCompilerId.c, 29](#)
 - [CMakeCXXCompilerId.cpp, 33](#)
- [info_language_standard_default](#)
 - [CMakeCCompilerId.c, 30](#)
 - [CMakeCXXCompilerId.cpp, 33](#)
- [info_platform](#)
 - [CMakeCCompilerId.c, 30](#)
 - [CMakeCXXCompilerId.cpp, 33](#)
- [LICENSE.MD, 33](#)
- [main](#)
 - [CMakeCCompilerId.c, 29](#)
 - [CMakeCXXCompilerId.cpp, 32](#)
 - [main.cpp, 34](#)
- [main.cpp, 33](#)
 - [main, 34](#)
- [operator<<](#)

- ShapeLib, [13](#)
- p1
 - ShapeLib::Rectangle, [22](#)
- p2
 - ShapeLib::Rectangle, [23](#)
- PLATFORM_ID
 - CMakeCCompilerId.c, [28](#)
 - CMakeCXXCompilerId.cpp, [32](#)
- Point, [15](#)
 - horizontalDistance, [16](#)
 - Point, [16](#)
 - verticalDistance, [16](#)
 - x, [17](#)
 - y, [17](#)
- point.cpp, [36](#)
- point.h, [34](#)
- printInfo
 - ShapeLib::Rectangle, [21](#)
 - ShapeLib::Shape, [24](#)
- README.md, [34](#)
- recalculateDim
 - ShapeLib::Rectangle, [21](#)
- Rectangle
 - ShapeLib::Rectangle, [19](#)
- rectangle.cpp, [36](#)
- rectangle.h, [34](#), [35](#)
- setP1
 - ShapeLib::Rectangle, [21](#)
- setP2
 - ShapeLib::Rectangle, [22](#)
- Shape
 - ShapeLib::Shape, [24](#)
- shape.cpp, [37](#)
- shape.h, [36](#)
- ShapeLib, [13](#)
 - operator<<, [13](#)
- ShapeLib::Rectangle, [18](#)
 - dimValid, [22](#)
 - getHeight, [20](#)
 - getP1, [20](#)
 - getP2, [20](#)
 - getWidth, [21](#)
 - height, [22](#)
 - p1, [22](#)
 - p2, [23](#)
 - printInfo, [21](#)
 - recalculateDim, [21](#)
 - Rectangle, [19](#)
 - setP1, [21](#)
 - setP2, [22](#)
 - width, [23](#)
- ShapeLib::Shape, [23](#)
 - getID, [24](#)
 - ID, [25](#)
 - printInfo, [24](#)
 - Shape, [24](#)
- STRINGIFY
 - CMakeCCompilerId.c, [29](#)
 - CMakeCXXCompilerId.cpp, [32](#)
- STRINGIFY_HELPER
 - CMakeCCompilerId.c, [29](#)
 - CMakeCXXCompilerId.cpp, [32](#)
- verticalDistance
 - Point, [16](#)
- width
 - ShapeLib::Rectangle, [23](#)
- x
 - Point, [17](#)
- y
 - Point, [17](#)