ShapeLib

Generated by Doxygen 1.9.3

Chapter 1

AK5PC - Shape 2D

Demonstration project for AK5PC course.

1.1 What you may find in this project about

Simplified demonstration of selected C++ features:

- Classes, abstract classes and basic inheritance
- Class members, member methods, functions and pure virtual functions
- Namespaces
- Simple constructors and constructor delegation
- Function overriding
- Simple operator overriding
- References and their usage
- Basic output stream usage
- etc.

1.2 What you DON'T find in this project

Because this project is most about C++ syntax and fundamental principles and is provided as supplemental material for the C++ classes, it's not focused on following things

- C++ best practices descriptions and demonstrations
- Application architecture recommendations
- Optimization guides

1.3 Project structure

The project is composed by CMake tool usage and is separated into two separate project. The first project is focused on ShapeLib and is placed with corresponding folder. It is configured as a shared library. The second project contains only short demonstration of the library written in main.cpp. The file and configuration is placed in root folder.

```
..
.
+-- doc # project documentation
|   +-- output
|   |   +-- html        # generated documentation in HTML format
|   |   +-- latex       # generated documentation in LaTex format (pdf)
|   +-- proj            # Doxygen project configuration
|
+-- ShapeLib            # the ShapeLib library project
|   +-- include         # header files
|   +-- src             # source files
|   +-- CMakeLists.txt  # library project configuration
|
+-- main.cpp        # the main demonstration file
+-- CMakeLists.txt  # the configuration for demonsration project
```

Chapter 2

LICENSE

Chapter 3

Namespace Index

3.1 Namespace List

Here is a list of all namespaces with brief descriptions:

Chapter 4

Hierarchical Index

4.1 Class Hierarchy

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

Chapter 5

Class Index

5.1 Class List

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

Chapter 6

File Index

6.1 File List

Here is a list of all files with brief descriptions:

Chapter 7

Namespace Documentation

7.1 ShapeLib Namespace Reference

Classes

- class Circle

  Simple class representing one Circle.
- class Point

  Simple class representing one point in cartesian coordinates.
- 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, Circle &circle)

  Stream operator for Circle class.
- std::ostream & operator<< (std::ostream &stream, const Point &point)

  Stream operator for the Point class.
- std::ostream & operator<< (std::ostream &stream, Rectangle &rect)

  Stream operator for Rectangle class.

7.1.1 Function Documentation

7.1.1.1 operator<<() [1/3]

```
std::ostream & ShapeLib::operator<< (
            std::ostream & stream,
            Circle & circle )
```

Stream operator for Circle class.

Parameters

stream	output stream
circle	the Circle object to be printed out

Returns

the original stream

7.1.1.2 operator<<() [2/3]

```
std::ostream & ShapeLib::operator<< (
            std::ostream & stream,
            const Point & point )
```

Stream operator for the Point class.

Parameters

stream	output stream
point	the Point object to be printed out

Returns

the original stream

7.1.1.3 operator<<() [3/3]

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

Stream operator for Rectangle class.

Parameters

stream	output stream
rect	the Rectangle object to be printed out

Returns

the original output stream

Chapter 8

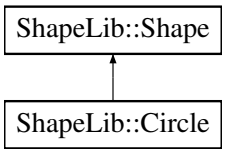
Class Documentation

8.1 ShapeLib::Circle Class Reference

Simple class representing one Circle.

`#include <circle.h>`

Inheritance diagram for ShapeLib::Circle:



Public Member Functions

- Circle (int ID, int centerX, int centerY, int radius)

  The circle constructor.
- Circle (int ID, const Point &center, int radius)

  The circle constructor.
- void setCenterPoint (const Point &centerPoint)

  Setter for the center point.
- void setRadius (int radius)

  Setter for the radius. It also invalidate internal dimensions.
- void printInfo () const override

  Print out tha basic circle info (center point and radius)
- const Point & getCenterPoint () const

  Getter for the center point.
- int getRadius () const

  Getter for the radius dimension.
- double getBorderLength ()

  This function returns the length of circle border. If the dimension is not valid, it is recalculated at first.
- double getCircleVolume ()

  this function returns the volume of circle area. If the dimensions is not valid, it is recalculated at first.

Protected Member Functions

- void recalculateDim ()

  Recalculates the circle border length and area volume.

Protected Attributes

- Point centerPoint

  Point defining the center of the circle.
- int radius

  Dimension of circle radius.
- double borderLength = 0

  Temporary storage of circle border length. Don't have to be correct!!!!
- double circleVolume = 0

  Temporary storage of circle area volume. Don't have to be correct!!!!
- bool dimsValid = false

  Information if circle area volume and border length is valido or not.

8.1.1 Detailed Description

Simple class representing one Circle.

The circle is defined by a centre point and radius.

8.1.2 Constructor & Destructor Documentation

8.1.2.1 Circle() [1/2]

```
ShapeLib::Circle::Circle (
            int ID,
            int centerX,
            int centerY,
            int radius )  [inline]
```

The circle 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

ID	Circle ID
centerX	The X coordinate of circle center
centerY	The Y coordinate of circle center
radius	The circle radius

8.1.2.2 Circle() [2/2]

```
ShapeLib::Circle::Circle (
            int ID,
            const Point & center,
            int radius )  [inline]
```

The circle constructor.

This constructor stores all Circle's internal members as well as members defined in Shape class. It also calls constructor of Shape class. Finally, it recalculates the dimensions of the rectangle.

See also

recalculateDim()

Parameters

ID	Circle ID
center	Circle center point

See also

Point

Parameters

radius	Circle radius

8.1.3 Member Function Documentation

8.1.3.1 getBorderLength()

```
double ShapeLib::Circle::getBorderLength ( )
```

This function returns the length of circle border. If the dimension is not valid, it is recalculated at first.

See also

recalculateDim()

Returns

current length of circle border.

8.1.3.2 getCenterPoint()

```
const Point & ShapeLib::Circle::getCenterPoint ( ) const  [inline]
```

Getter for the center point.

Returns

Center point

8.1.3.3 getCircleVolume()

```
double ShapeLib::Circle::getCircleVolume ( )
```

this function returns the volume of circle area. If the dimensions is not valid, it is recalculated at first.

See also

recalculateDim()

Returns

8.1.3.4 getRadius()

```
int ShapeLib::Circle::getRadius ( ) const  [inline]
```

Getter for the radius dimension.

Returns

Size of the circle's radius

8.1.3.5 printInfo()

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

Print out tha basic circle info (center point and radius)

Implements ShapeLib::Shape.

8.1.3.6 recalculateDim()

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

Recalculates the circle border length and area volume.

Based on radiusthis function recalculates the circle are volume and border lenght and store it in internal temporary variables.

See also

borderLength

circleVolume

8.1.3.7 setCenterPoint()

```
void ShapeLib::Circle::setCenterPoint (
            const Point & centerPoint )
```

Setter for the center point.

Parameters

centerPoint	Point defining the circle center

8.1.3.8 setRadius()

```
void ShapeLib::Circle::setRadius (
            int radius )
```

Setter for the radius. It also invalidate internal dimensions.

Parameters

radius	the circle radius

8.1.4 Member Data Documentation

8.1.4.1 borderLength

```
double ShapeLib::Circle::borderLength = 0  [protected]
```

Temporary storage of circle border length. Don't have to be correct!!!!

See also

dimsValid

8.1.4.2 centerPoint

```
Point ShapeLib::Circle::centerPoint  [protected]
```

Point defining the center of the circle.

8.1.4.3 circleVolume

```
double ShapeLib::Circle::circleVolume = 0  [protected]
```

Temporary storage of circle area volume. Don't have to be correct!!!!

See also

dimsValid

8.1.4.4 dimsValid

```
bool ShapeLib::Circle::dimsValid = false  [protected]
```

Information if circle area volume and border length is valido or not.

See also

recalculateDim()

8.1.4.5 radius

```
int ShapeLib::Circle::radius  [protected]
```

Dimension of circle radius.

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

- circle.h
- circle.cpp

8.2 ShapeLib::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

8.2.1 Detailed Description

Simple class representing one point in cartesian coordinates.

8.2.2 Constructor & Destructor Documentation

8.2.2.1 Point()

```
ShapeLib::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

x	coordinate
y	coordinate

8.2.3 Member Function Documentation

8.2.3.1 horizontalDistance()

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

Static function for horizontal distance of two point estimation.

See also

Point

Parameters

p1	first input point
p2	second input point

Returns

horizontal distance between provided pints

8.2.3.2 verticalDistance()

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

Static function for vertical distance of two point estimation.

See also

Point

Parameters

p1	first input point
p2	second input point

Returns

vertical distance between provided points

8.2.4 Member Data Documentation

8.2.4.1 x

```
int ShapeLib::Point::x
```

coordinate X

8.2.4.2 y

```
int ShapeLib::Point::y
```

coordinate Y

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

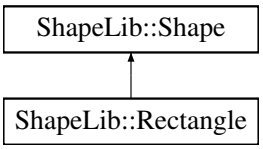
- point.h
- point.cpp

8.3 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

  Print out basic rectangle info (ID, P2 and P2)
- 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

  The first point in 2D space.
- Point p2

  The second 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.

8.3.1 Detailed Description

Simple class representing a rectangle defined by two points.

See also

Point

8.3.2 Constructor & Destructor Documentation

8.3.2.1 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

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

8.3.2.2 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 dimensions of the rectangle.

See also

recalculateDim()

Parameters

id	Rectangle ID
p1	First point defining the rectangle.
p2	Second point defining the rectangle.

8.3.3 Member Function Documentation

8.3.3.1 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.

8.3.3.2 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

8.3.3.3 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

8.3.3.4 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.

8.3.3.5 printInfo()

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

Print out basic rectangle info (ID, P2 and P2)

Implements ShapeLib::Shape.

8.3.3.6 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

8.3.3.7 setP1()

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

Setter for the first point of the rectangle.

Parameters

p1	point defining the rectangle.

8.3.3.8 setP2()

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

Setter for the second point of the rectangle.

Parameters

p2	point defining the rectangle.

8.3.4 Member Data Documentation

8.3.4.1 dimValid

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

Information if width and height is correct or not.

See also

recalculateDim()

8.3.4.2 height

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

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

See also

dimValid

8.3.4.3 p1

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

The first point in 2D space.

8.3.4.4 p2

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

The second point in 2D space.

8.3.4.5 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:

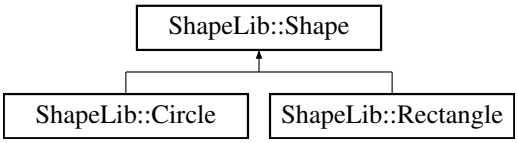
- rectangle.h
- rectangle.cpp

8.4 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)

  Constructor of Shape class.
- 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

8.4.1 Detailed Description

Interface for any 2D shape in this library.

This function implements the basic ID management as well as virtual function printInfo().

8.4.2 Constructor & Destructor Documentation

8.4.2.1 Shape()

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

Constructor of Shape class.

Parameters

id	an ID of the shape

8.4.3 Member Function Documentation

8.4.3.1 getID()

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

Getter for shape's ID.

Returns

the ID of the shape

8.4.3.2 printInfo()

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

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

Implemented in ShapeLib::Circle, and ShapeLib::Rectangle.

8.4.4 Member Data Documentation

8.4.4.1 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"
#include "circle.h"
```

Functions

- int main ()

9.4.1 Function Documentation

9.4.1.1 main()

```
int main ( )
```

9.5 README.md File Reference

9.6 circle.h File Reference

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

Classes

- class ShapeLib::Circle

  Simple class representing one Circle.

Namespaces

- namespace ShapeLib

Functions

- std::ostream & ShapeLib::operator<< (std::ostream &stream, Circle &circle)

  Stream operator for Circle class.

9.7 circle.h

Go to the documentation of this file.

```
1 //
2 // Created by Peter Janků on 10.10.2022.
3 //
4
5 #ifndef SHAPE2D_CIRCLE_H
6 #define SHAPE2D_CIRCLE_H
7
8 #include "shape.h"
9 #include "point.h"
10
11 namespace ShapeLib {
17     class Circle : public Shape {
18
19     public:
29         Circle(int ID, int centerX, int centerY, int radius) : Circle(ID, Point(centerX, centerY),
       radius) {}
30
41         Circle(int ID, const Point &center, int radius) : Shape(ID), centerPoint(center), radius(radius)
       {}
42
43         void setCenterPoint(const Point &centerPoint);
44
45         void setRadius(int radius);
46
47         void printInfo() const override;
48
53         const Point &getCenterPoint() const {
54             return centerPoint;
55         }
56
61         int getRadius() const {
62             return radius;
63         }
64         double getBorderLength();
65         double getCircleVolume();
66
67     protected:
68
69         void recalculateDim();
70
71         Point centerPoint;
72         int radius;
73         double borderLength = 0;
74         double circleVolume = 0;
75         bool dimsValid = false;
76     };
77
78     std::ostream &operator<<(std::ostream &stream, Circle &circle);
79
80 } // ShapeLib
81
82 #endif //SHAPE2D_CIRCLE_H
```

9.8 point.h File Reference

```
#include <iostream>
```

Classes

- class ShapeLib::Point

  Simple class representing one point in cartesian coordinates.

Namespaces

- namespace ShapeLib

Functions

- std::ostream & ShapeLib::operator<< (std::ostream &stream, const Point &point)

  Stream operator for the Point class.

9.9 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
8 #include <iostream>
9
10 namespace ShapeLib {
14     class Point {
15     public:
16
24         Point(int x, int y) : x(x), y(y) {}
25
26         static int horizontalDistance(const Point &p1, const Point &p2);
27
28         static int verticalDistance(const Point &p1, const Point &p2);
29
30         int x;
31         int y;
32     };
33
34     std::ostream &operator«(std::ostream& stream, const Point &point);
35
36 }
37
38 #endif //SHAPE2D_POINT_H
```

9.10 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)

  Stream operator for Rectangle class.

9.11 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 {
16     class Rectangle : public Shape {
17     public:
18
29         Rectangle(int id, int x1, int y1, int x2, int y2) : Rectangle(id, Point(x1, y1), Point(x2, y2))
       {}
30
40         Rectangle(int id, const Point &p1, const Point &p2) : Shape(id), p1(p1), p2(p2) {
       recalculateDim(); }
41
42         void printInfo() const override;
43
44         int getWidth();
45
46         int getHeight();
47
52         const Point &getP1() const { return p1; }
53
58         const Point &getP2() const { return p2; }
59
60         void setP1(const Point &p1);
61
62         void setP2(const Point &p2);
63
64     protected:
65
66         void recalculateDim();
67         Point p1;
68         Point p2;
69         mutable int width;
70         mutable int height;
71         mutable bool dimValid = false;
74     };
75
76
77     std::ostream &operator«(std::ostream &stream, Rectangle &rect);
78 }
79
80
81 #endif //SHAPE2D_RECTANGLE_H
```

9.12 shape.h File Reference

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

Classes

- class ShapeLib::Shape

  Interface for any 2D shape in this library.

Namespaces

- namespace ShapeLib

9.13 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 {
16     class Shape {
17     public:
18
23         explicit Shape(int id) : ID(id) {}
24
28         virtual void printInfo() const = 0;
29
34         int getID() const { return ID; }
35
36     protected:
37         int ID;
38     };
39
40 } // ShapeLib
41
42 #endif //SHAPE2D_SHAPE_H
```

9.14 circle.cpp File Reference

```
#include <cmath>
#include <iostream>
#include "circle.h"
```

Namespaces

- namespace ShapeLib

Functions

- std::ostream & ShapeLib::operator<< (std::ostream &stream, Circle &circle)

  Stream operator for Circle class.

9.15 point.cpp File Reference

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

Namespaces

- namespace ShapeLib

Functions

- std::ostream & ShapeLib::operator<< (std::ostream &stream, const Point &point)

  Stream operator for the Point class.

9.16 rectangle.cpp File Reference

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

9.17 shape.cpp File Reference

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

Namespaces

- namespace ShapeLib

Index