ShapeLib

Generated by Doxygen 1.9.3

Chapter 1

AK5PC - Shape 2D

Demonstration project for AK5PC course.

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 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

x	coordinate
y	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

p1	first input point
p2	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

p1	first input point
p2	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:

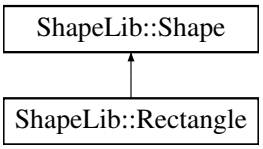
- point.h
- point.cpp

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

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

8.2.2.2 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

id	
p1	
p2	

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

p1	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

p2	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:

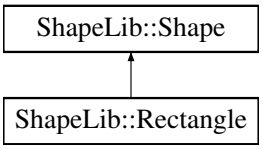
- rectangle.h
- rectangle.cpp

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]
```

@brieaf Constructor of Shape class

Parameters

id	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 {
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     protected:
61
62         void recalculateDim();
63
64         Point p1;
65     public:
66         void setP1(const Point &p1);
67
68         void setP2(const Point &p2);
69
70     protected:
72         Point p2;
73         mutable int width;
74         mutable int height;
75         mutable bool dimValid = false;
78     };
79
80
81     std::ostream &operator«(std::ostream &stream, Rectangle &rect);
82 }
83
84
85 #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 {
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
```

9.12 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