Navigation: Up, Table of Contents, Bibliography, Index, Title Page

Class

CGAL::Point_2<Kernel>

Definition

An object of the class *Point_2*<*Kernel*> is a point in the two-dimensional Euclidean plane $\mathbb{E}^2$.

Remember that *Kernel*::*RT* and *Kernel*::*FT* denote a RingNumberType and a FieldNumberType, respectively. For the kernel model *Cartesian*<*T*>, the two types are the same. For the kernel model *Homogeneous*<*T*>, *Kernel*::*RT* is equal to *T*, and *Kernel*::*FT* is equal to *Quotient*<*T*>.

Types

Point_2<*Kernel*>::*Cartesian_const_iterator*

An iterator for enumerating the Cartesian coordinates of a point.

Creation

Point_2<*Kernel*> *p* (*Origin ORIGIN*);

introduces a variable *p* with Cartesian coordinates *(0,0)*.

Point_2<*Kernel*> *p* (*Kernel*::*RT hx*, *Kernel*::*RT hy*, *Kernel*::*RT hw = RT(1)*);

introduces a point *p* initialized to *(hx/hw,hy/hw)*.
Precondition: hw ≠ *Kernel*::*RT(0)*

Operations

bool	*p.operator== (q)*	Test for equality. Two points are equal, iff their *x* and *y* coordinates are equal. The point can be compared with *ORIGIN*.
bool	*p.operator!= (q)*	Test for inequality. The point can be compared with *ORIGIN*.

There are two sets of coordinate access functions, namely to the homogeneous and to the Cartesian coordinates. They can be used independently from the chosen kernel model.

Kernel::*RT*	*p.hx ()*	returns the homogeneous *x* coordinate.
Kernel::*RT*	*p.hy ()*	returns the homogeneous *y* coordinate.

Kernel::RT	*p.hw ()*	returns the homogenizing coordinate.

Note that you do not loose information with the homogeneous representation, because the FieldNumberType is a quotient.

Kernel::FT	*p.x ()*	returns the Cartesian *x* coordinate, that is *hx/hw*.
Kernel::FT	*p.y ()*	returns the Cartesian *y* coordinate, that is *hy/hw*.

The following operations are for convenience and for compatibility with higher dimensional points. Again they come in a Cartesian and in a homogeneous flavor.

Kernel::RT	*p.homogeneous (int i)*	returns the i'th homogeneous coordinate of *p*, starting with 0. *Precondition:* $0 \leq i \leq 2$.
Kernel::FT	*p.cartesian (int i)*	returns the i'th Cartesian coordinate of *p*, starting with 0. *Precondition:* $0 \leq i \leq 1$.
Kernel::FT	*p.operator[] (int i)*	returns *cartesian(i)*. *Precondition:* $0 \leq i \leq 1$.
Cartesian_const_iterator	*p.cartesian_begin ()*	returns an iterator to the Cartesian coordinates of *p*, starting with the 0th coordinate.
Cartesian_const_iterator	*p.cartesian_end ()*	returns an off the end iterator to the Cartesian coordinates of *p*.
int	*p.dimension ()*	returns the dimension (the constant 2).
Bbox_2	*p.bbox ()*	returns a bounding box containing *p*. Note that bounding boxes are not parameterized with whatsoever.
Point_2<Kernel>	*p.transform (Aff_transformation_2<Kernel> t)*	returns the point obtained by applying *t* on *p*.

Operators

The following operations can be applied on points:

bool	*operator< (p, q)*	returns true iff *p* is lexicographically smaller than *q*, i.e. either if *p.x() < q.x()* or if *p.x() == q.x()* and *p.y() < q.y()*.
bool	*operator> (p, q)*	returns true iff *p* is lexicographically greater than *q*.
bool	*operator<= (p, q)*	returns true iff *p* is lexicographically smaller or equal to *q*.
bool	*operator>= (p, q)*	returns true iff *p* is lexicographically greater or equal to *q*.
Vector_2<Kernel>	*operator- (p, q)*	returns the difference vector between *q* and *p*. You can substitute *ORIGIN* for either *p* or *q*, but not for both.
Point_2<Kernel>	*operator+ (p, Vector_2<Kernel> v)*	returns the point obtained by translating *p* by the vector *v*.
Point_2<Kernel>	*operator- (p, Vector_2<Kernel> v)*	returns the point obtained by translating *p* by the vector -*v*.

Example

The following declaration creates two points with Cartesian double coordinates.

```
Point_2< Cartesian<double> > p, q(1.0, 2.0);
```

The variable `p` is uninitialized and should first be used on the left hand side of an assignment.

```
p = q;
std::cout << p.x() << "  " << p.y() << std::endl;
```

See Also

Kernel::Point_2

Next: *Ray_2<Kernel>*

Navigation: Table of Contents, Bibliography, Index, Title Page

The CGAL Project . Tue, December 21, 2004 .