

sikidom

1.0.0

Készítette Doxygen 1.10.0

1. Hierarchikus mutató	1
1.1. Osztályhierarchia	1
2. Osztálymutató	3
2.1. Osztálylista	3
3. Fájlmutató	5
3.1. Fájllista	5
4. Osztályok dokumentációja	7
4.1. Tomb< T >::const_iterator osztályreferencia	7
4.1.1. Részletes leírás	7
4.1.2. Konstruktork és destruktorok dokumentációja	8
4.1.2.1. const_iterator()	8
4.1.3. Tagfüggvények dokumentációja	9
4.1.3.1. operator!=(())	9
4.1.3.2. operator*()	9
4.1.3.3. operator++()	9
4.1.4. Adattagok dokumentációja	10
4.1.4.1. idx	10
4.1.4.2. p	10
4.2. Haromszog osztályreferencia	10
4.2.1. Részletes leírás	11
4.2.2. Konstruktork és destruktorok dokumentációja	12
4.2.2.1. Haromszog() [1/2]	12
4.2.2.2. Haromszog() [2/2]	12
4.2.2.3. ~Haromszog()	12
4.2.3. Tagfüggvények dokumentációja	12
4.2.3.1. Kivul()	12
4.2.3.2. Rajtavan()	12
4.2.3.3. Read()	13
4.2.3.4. Terulet()	13
4.2.3.5. Write()	13
4.2.4. Barát és kapcsolódó függvények dokumentációja	14
4.2.4.1. operator<<	14
4.3. Tomb< T >::iterator osztályreferencia	14
4.3.1. Részletes leírás	14
4.3.2. Konstruktork és destruktorok dokumentációja	14
4.3.2.1. iterator()	14
4.3.3. Tagfüggvények dokumentációja	15
4.3.3.1. operator!=(())	15
4.3.3.2. operator*()	15
4.3.3.3. operator++()	15

4.3.4.	Adattagok dokumentációja	16
4.3.4.1.	idx	16
4.3.4.2.	p	16
4.4.	Kor osztályreferencia	16
4.4.1.	Részletes leírás	17
4.4.2.	Konstruktorok és destruktorok dokumentációja	18
4.4.2.1.	Kor() [1/2]	18
4.4.2.2.	Kor() [2/2]	18
4.4.2.3.	~Kor()	18
4.4.3.	Tagfüggvények dokumentációja	18
4.4.3.1.	Kivul()	18
4.4.3.2.	Rajtavan()	18
4.4.3.3.	Read()	19
4.4.3.4.	Terulet()	19
4.4.3.5.	Write()	19
4.4.4.	Barát és kapcsolódó függvények dokumentációja	20
4.4.4.1.	operator<<	20
4.5.	Negyzet osztályreferencia	20
4.5.1.	Részletes leírás	21
4.5.2.	Konstruktorok és destruktorok dokumentációja	21
4.5.2.1.	Negyzet() [1/2]	21
4.5.2.2.	Negyzet() [2/2]	21
4.5.2.3.	~Negyzet()	21
4.5.3.	Tagfüggvények dokumentációja	22
4.5.3.1.	Kivul()	22
4.5.3.2.	Rajtavan()	22
4.5.3.3.	Read()	22
4.5.3.4.	Terulet()	23
4.5.3.5.	Write()	23
4.5.4.	Barát és kapcsolódó függvények dokumentációja	23
4.5.4.1.	operator<<	23
4.6.	Pont osztályreferencia	23
4.6.1.	Részletes leírás	24
4.6.2.	Konstruktorok és destruktorok dokumentációja	25
4.6.2.1.	Pont() [1/2]	25
4.6.2.2.	Pont() [2/2]	25
4.6.3.	Tagfüggvények dokumentációja	25
4.6.3.1.	dst()	25
4.6.3.2.	Forgat()	25
4.6.3.3.	getx()	26
4.6.3.4.	gety()	26
4.6.3.5.	Mozgat()	26

4.6.3.6.	operator!=()	26
4.6.3.7.	operator+()	27
4.6.3.8.	operator-()	27
4.6.3.9.	operator=()	27
4.6.3.10.	operator==()	28
4.6.3.11.	setx()	28
4.6.3.12.	sety()	28
4.6.4.	Barát és kapcsolódó függvények dokumentációja	29
4.6.4.1.	operator<<	29
4.6.4.2.	operator>>	29
4.6.5.	Adattagok dokumentációja	30
4.6.5.1.	x	30
4.6.5.2.	y	30
4.7.	Sikidom osztályreferencia	30
4.7.1.	Részletes leírás	31
4.7.2.	Konstruktorok és destruktorok dokumentációja	31
4.7.2.1.	Sikidom() [1/2]	31
4.7.2.2.	Sikidom() [2/2]	32
4.7.2.3.	~Sikidom()	32
4.7.3.	Tagfüggvények dokumentációja	32
4.7.3.1.	createSikidom()	32
4.7.3.2.	Forgat()	32
4.7.3.3.	getkp()	33
4.7.3.4.	getp()	33
4.7.3.5.	Kivul()	33
4.7.3.6.	Mozgat()	33
4.7.3.7.	operator=()	34
4.7.3.8.	Rajtavan()	34
4.7.3.9.	Read()	34
4.7.3.10.	Tartalmazza()	34
4.7.3.11.	Terulet()	34
4.7.3.12.	Write()	34
4.7.4.	Barát és kapcsolódó függvények dokumentációja	35
4.7.4.1.	operator<<	35
4.7.4.2.	operator>>	36
4.7.5.	Adattagok dokumentációja	36
4.7.5.1.	kp	36
4.7.5.2.	p	36
4.8.	Tomb< T > osztálysablon-referencia	37
4.8.1.	Részletes leírás	38
4.8.2.	Konstruktorok és destruktorok dokumentációja	38
4.8.2.1.	Tomb()	38

4.8.2.2.	<code>~Tomb()</code>	38
4.8.3.	Tagfüggvények dokumentációja	38
4.8.3.1.	<code>at()</code> [1/2]	38
4.8.3.2.	<code>at()</code> [2/2]	39
4.8.3.3.	<code>begin()</code> [1/2]	39
4.8.3.4.	<code>begin()</code> [2/2]	39
4.8.3.5.	<code>capacity()</code>	39
4.8.3.6.	<code>end()</code> [1/2]	40
4.8.3.7.	<code>end()</code> [2/2]	40
4.8.3.8.	<code>find()</code>	40
4.8.3.9.	<code>operator[]()</code> [1/2]	40
4.8.3.10.	<code>operator[]()</code> [2/2]	41
4.8.3.11.	<code>push_back()</code>	41
4.8.3.12.	<code>resize()</code>	41
4.8.3.13.	<code>size()</code>	42
4.8.4.	Adattagok dokumentációja	42
4.8.4.1.	<code>currentCapacity</code>	42
4.8.4.2.	<code>currentSize</code>	42
4.8.4.3.	<code>t</code>	42
5.	Fájlok dokumentációja	43
5.1.	<code>src/pont.cpp</code> fájlreferencia	43
5.1.1.	Függvények dokumentációja	43
5.1.1.1.	<code>dst()</code>	43
5.1.1.2.	<code>operator<<()</code>	44
5.1.1.3.	<code>operator>>()</code>	44
5.2.	<code>src/pont.h</code> fájlreferencia	44
5.2.1.	Függvények dokumentációja	45
5.2.1.1.	<code>dst()</code>	45
5.3.	<code>pont.h</code>	45
5.4.	<code>src/sikidom.cpp</code> fájlreferencia	46
5.4.1.	Függvények dokumentációja	46
5.4.1.1.	<code>IsOnTriangle()</code>	46
5.4.1.2.	<code>operator<<()</code>	47
5.4.1.3.	<code>operator>>()</code>	47
5.5.	<code>src/sikidom.h</code> fájlreferencia	47
5.5.1.	Függvények dokumentációja	48
5.5.1.1.	<code>IsOnTriangle()</code>	48
5.6.	<code>sikidom.h</code>	48
5.7.	<code>src/tomb.hpp</code> fájlreferencia	50
5.8.	<code>tomb.hpp</code>	50
Tárgymutató		53

1. fejezet

Hierarchikus mutató

1.1. Osztályhierarchia

Majdnem (de nem teljesen) betűrendbe szedett leszármazási lista:

Tomb< T >::const_iterator	7
Tomb< T >::iterator	14
Pont	23
Sikidom	30
Haromszog	10
Kor	16
Negyzet	20
Tomb< T >	37

2. fejezet

Osztálymutató

2.1. Osztálylista

Az összes osztály, struktúra, unió és interfész listája rövid leírásokkal:

Tomb< T >::const_iterator	
Const_iterator osztály. Az const_iterator osztály a Tomb osztály const_iteratora	7
Haromszog	
Haromszog osztály. A Haromszog osztály a Sikidom leszármazottja, mely egy háromszög síkidomot reprezentál, középpontjával és egy csúcsával	10
Tomb< T >::iterator	
Iterator osztály. Az iterator osztály a Tomb osztály iteratora	14
Kor	
Kor osztály. A Kor osztály a Sikidom leszármazottja, mely egy kör síkidomot reprezentál, középpontjával és egy csúcsával	16
Negyzet	20
Pont	
Pont osztály A pontokat a síkon tárolja, x és y koordinátákkal	23
Sikidom	
Sikidom osztály. A sikidom osztály egy absztrakt osztály, melynek leszármazottai a különböző (szabályos) síkidomokat reprezentálják, középpontjukkal és egy csúcsukkal	30
Tomb< T >	
Generikus Dinamikus tömb osztály. A Tomb osztály egy dinamikus tömböt reprezentál, melynek mérete a felhasználó által nem korlátozott	37

3. fejezet

Fájlmutató

3.1. Fájllista

Az összes fájl listája rövid leírásokkal:

src/ pont.cpp	43
src/ pont.h	44
src/ sikidom.cpp	46
src/ sikidom.h	47
src/ tomb.hpp	50

4. fejezet

Osztályok dokumentációja

4.1. Tomb< T >::const_iterator osztályreferencia

`const_iterator` osztály. Az `const_iterator` osztály a `Tomb` osztály `const_iterator`-a.

```
#include <tomb.hpp>
```

Publikus tagfüggvények

- `const_iterator` (`const T *p=nullptr, size_t idx=0`)
`const_iterator` konstruktor.
- `const T & operator* () const`
`operator` függvény.*
- `const_iterator & operator++ ()`
`operator++` függvény.
- `bool operator!= (const const_iterator &other) const`
`operator!=` függvény.

Privát attribútumok

- `const T * p`
Az `const_iterator` aktuális elemre mutató pointer.
- `size_t idx`
Az `const_iterator` aktuális indexe.

4.1.1. Részletes leírás

```
template<class T>  
class Tomb< T >::const_iterator
```

`const_iterator` osztály. Az `const_iterator` osztály a `Tomb` osztály `const_iterator`-a.

4.1.2. Konstruktorkok és destruktorkok dokumentációja

4.1.2.1. `const_iterator()`

```
template<class T >
Tomb< T >::const_iterator::const_iterator (
    const T * p = nullptr,
    size_t idx = 0 ) [inline]
```

`const_iterator` konstruktora.

Paraméterek

<i>p</i>	Az <code>const_iterator</code> aktuális elemre mutató pointerre.
<i>idx</i>	Az <code>const_iterator</code> aktuális indexe.

4.1.3. Tagfüggvények dokumentációja

4.1.3.1. operator"!=()

```
template<class T >
bool Tomb< T >::const_iterator::operator!= (
    const const_iterator & other ) const [inline]
```

operator!= függvény.

Paraméterek

<i>other</i>	A másik <code>const_iterator</code> .
--------------	---------------------------------------

Visszatérési érték

Igaz, ha a két `const_iterator` nem egyezik meg, egyébként hamis.

4.1.3.2. operator*()

```
template<class T >
const T & Tomb< T >::const_iterator::operator* ( ) const [inline]
```

operator* függvény.

Visszatérési érték

Az `const_iterator` aktuális elemre mutató referencia.

4.1.3.3. operator++()

```
template<class T >
const_iterator & Tomb< T >::const_iterator::operator++ ( ) [inline]
```

operator++ függvény.

Visszatérési érték

Az `const_iterator` az aktuális elem után mutató referencia.

4.1.4. Adattagok dokumentációja

4.1.4.1. idx

```
template<class T >
size_t Tomb< T >::const_iterator::idx [private]
```

Az `const_iterator` aktuális indexe.

4.1.4.2. p

```
template<class T >
const T* Tomb< T >::const_iterator::p [private]
```

Az `const_iterator` aktuális elemre mutató pointer.

Ez a dokumentáció az osztályról a következő fájl alapján készült:

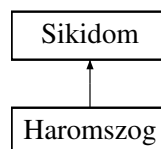
- `src/tomb.hpp`

4.2. Haromszog osztályreferencia

`Haromszog` osztály. A `Haromszog` osztály a `Sikidom` leszármazottja, mely egy háromszög síkidomot reprezentál, középpontjával és egy csúcsával.

```
#include <sikidom.h>
```

A `Haromszog` osztály származási diagramja:



Publikus tagfüggvények

- `Haromszog (Pont _kp=Pont(0, 0), Pont _p=Pont(0, 0))`
- `Haromszog (const Haromszog &_h)`
- `double Terulet ()` const override
Haromszog területét számoló függvény. A kiszámítás módja: $R^2 \cdot 3 \cdot \sqrt{3} / 4$.
- `bool Rajtavan (const Pont &)` const override
Haromszog Rajtavan függvény. A függvény eldönti, hogy egy pont rajta van-e a háromszögön.
- `void Write (std::ostream &os)` const override
Haromszog író függvény. A függvény kiírja a háromszög adatait a megadott output streambe.
- `void Read (std::istream &is)` override
Haromszog beolvasó függvény. A függvény beolvassa a háromszög adatait a megadott input streamből.
- `bool Kivul (const std::size_t r)` const override
Haromszog Kivul függvény. Eldönti, hogy a háromszög kívül van-e egy adott sugarú, origó középpontú körön.
- `~Haromszog ()` override

Publikus tagfüggvények a(z) **Sikidom** osztályból származnak

- **Sikidom** (const **Sikidom** &)
- **Sikidom** & **operator=** (const **Sikidom** &)
Sikidom osztály egyenlőség operátora.
- **Pont** **getkp** () const
- **Pont** **getp** () const
- bool **Tartalmazza** (const int) const
- **Sikidom** & **Mozgat** (const int _x, const int _y)
Egy sokszög mozgatását teszi lehetővé.
- **Sikidom** & **Forgat** (const double, const **Pont** &)
Egy sokszög forgatását teszi lehetővé.
- virtual ~**Sikidom** ()=0
Sikidom osztály destruktora. nincs dinamikusan foglalt adattag, tehát nem kell semmit felszabadítani.

Barátok

- std::ostream & **operator<<** (std::ostream &, const **Haromszog** &)

További örökölt tagok

Statikus publikus tagfüggvények a(z) **Sikidom** osztályból származnak

- static **Sikidom** * **createSikidom** (const std::string &type)
Sikidom osztály statikus factory függvénye. A függvény létrehoz egy új síkidomot a megadott típus alapján.

Védett tagfüggvények a(z) **Sikidom** osztályból származnak

- **Sikidom** (**Pont** _kp=**Pont**(0, 0), **Pont** _p=**Pont**(0, 0))
Sikidom konstruktor.

Védett attribútumok a(z) **Sikidom** osztályból származnak

- **Pont** kp
Középpont.
- **Pont** p
A síkidom egy csúcsa.

4.2.1. Részletes leírás

Haromszog osztály. A **Haromszog** osztály a **Sikidom** leszármazottja, mely egy háromszög síkidomot reprezentál, középpontjával és egy csúcsával.

4.2.2. Konstruktorkorok és destruktorkorok dokumentációja

4.2.2.1. Haromszog() [1/2]

```
Haromszog::Haromszog (
    Pont _kp = Pont (0, 0),
    Pont _p = Pont (0, 0) ) [inline]
```

4.2.2.2. Haromszog() [2/2]

```
Haromszog::Haromszog (
    const Haromszog & _h ) [inline]
```

4.2.2.3. ~Haromszog()

```
Haromszog::~Haromszog ( ) [inline], [override]
```

4.2.3. Tagfüggvények dokumentációja

4.2.3.1. Kivul()

```
bool Haromszog::Kivul (
    const std::size_t r = 1 ) const [override], [virtual]
```

[Haromszog](#) Kivul függvény. Eldönti, hogy a háromszög kívül van-e egy adott sugarú, origo középpontú körön.

Paraméterek

<i>r</i>	a kör sugara.
----------	---------------

Visszatérési érték

true, ha a háromszög kívül van a körön, egyébként false.

Megvalósítja a következőket: [Sikidom](#).

4.2.3.2. Rajtavan()

```
bool Haromszog::Rajtavan (
    const Pont & P ) const [override], [virtual]
```

[Haromszog](#) Rajtavan függvény. A függvény eldönti, hogy egy pont rajta van-e a háromszögön.

Paraméterek

<i>P</i>	a vizsgált pont.
----------	------------------

Visszatérési érték

true, ha a pont rajta van a háromszögön, egyébként false.

Megvalósítja a következőket: [Sikidom](#).

4.2.3.3. Read()

```
void Háromszög::Read (
    std::istream & is ) [override], [virtual]
```

[Háromszög](#) beolvasó függvény. A függvény beolvassa a háromszög adatait a megadott input streamből.

Paraméterek

<i>is</i>	a bemeneti stream.
-----------	--------------------

Megvalósítja a következőket: [Sikidom](#).

4.2.3.4. Terület()

```
double Háromszög::Terület ( ) const [override], [virtual]
```

[Háromszög](#) területét számoló függvény. A kiszámítás módja: $R^2 * 3 * \text{sqrt}(3) / 4$.

Visszatérési érték

A háromszög területe.

Megvalósítja a következőket: [Sikidom](#).

4.2.3.5. Write()

```
void Háromszög::Write (
    std::ostream & os ) const [override], [virtual]
```

[Háromszög](#) kiíró függvény. A függvény kiírja a háromszög adatait a megadott output streambe.

Paraméterek

<i>os</i>	a kimeneti stream.
-----------	--------------------

Megvalósítja a következőket: [Sikidom](#).

4.2.4. Barát és kapcsolódó függvények dokumentációja

4.2.4.1. operator<<

```
std::ostream & operator<< (
    std::ostream & ,
    const Haromszog & ) [friend]
```

Ez a dokumentáció az osztályról a következő fájlok alapján készült:

- src/sikidom.h
- src/sikidom.cpp

4.3. Tomb< T >::iterator osztályreferencia

iterator osztály. Az iterator osztály a Tomb osztály iteratora.

```
#include <tomb.hpp>
```

Publikus tagfüggvények

- iterator (T *p=nullptr, size_t idx=0)
iterator konstruktora.
- T & operator* ()
operator függvény.*
- iterator & operator++ ()
operator++ függvény.
- bool operator!= (const iterator &other) const
operator!= függvény.

Privát attribútumok

- T * p
Az iterator aktuális elemre mutató pointer.
- size_t idx
Az iterator aktuális indexe.

4.3.1. Részletes leírás

```
template<class T>
class Tomb< T >::iterator
```

iterator osztály. Az iterator osztály a Tomb osztály iteratora.

4.3.2. Konstruktorkok és destruktorkok dokumentációja

4.3.2.1. iterator()

```
template<class T >
Tomb< T >::iterator::iterator (
    T * p = nullptr,
    size_t idx = 0 ) [inline]
```

iterator konstruktora.

Paraméterek

<i>p</i>	Az iterator aktuális elemre mutató pointer.
<i>idx</i>	Az iterator aktuális indexe.

4.3.3. Tagfüggvények dokumentációja

4.3.3.1. operator"!=()

```
template<class T >
bool Tomb< T >::iterator::operator!= (
    const iterator & other ) const [inline]
```

operator!= függvény.

Paraméterek

<i>other</i>	A másik iterator.
--------------	-------------------

Visszatérési érték

Igaz, ha a két iterator nem egyezik meg, egyébként hamis.

4.3.3.2. operator*()

```
template<class T >
T & Tomb< T >::iterator::operator* ( ) [inline]
```

operator* függvény.

Visszatérési érték

Az iterator aktuális elemre mutató referencia.

4.3.3.3. operator++()

```
template<class T >
iterator & Tomb< T >::iterator::operator++ ( ) [inline]
```

operator++ függvény.

Visszatérési érték

Az iterator az aktuális elem után mutató referencia.

4.3.4. Adattagok dokumentációja

4.3.4.1. idx

```
template<class T >
size_t Tomb< T >::iterator::idx [private]
```

Az iterator aktuális indexe.

4.3.4.2. p

```
template<class T >
T* Tomb< T >::iterator::p [private]
```

Az iterator aktuális elemre mutató pointer.

Ez a dokumentáció az osztályról a következő fájl alapján készült:

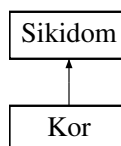
- [src/tomb.hpp](#)

4.4. Kor osztályreferencia

[Kor](#) osztály. A [Kor](#) osztály a [Sikidom](#) leszármazottja, mely egy kör síkidomot reprezentál, középpontjával és egy csúcsával.

```
#include <sikidom.h>
```

A [Kor](#) osztály származási diagramja:



Publikus tagfüggvények

- [Kor](#) ([Pont](#) _kp=[Pont](#)(0, 0), [Pont](#) _p=[Pont](#)(0, 0))
- [Kor](#) (const [Kor](#) &_k)
- double [Terulet](#) () const override
*[Kor](#) területét számoló függvény. A kiszamitas modja: $r^2 * \pi$.*
- bool [Rajtavan](#) (const [Pont](#) &) const override
[Kor](#) Rajtavan függvény. A függvény eldönti, hogy egy pont rajta van-e a körön.
- void [Write](#) (std::ostream &os) const override
[Kor](#) kíró függvény. A függvény kírja a kör adatait a megadott output streambe.
- void [Read](#) (std::istream &is) override
[Kor](#) beolvasó függvény. A függvény beolvassa a kör adatait a megadott input streamből.
- bool [Kivul](#) (const std::size_t r) const override
[Kor](#) kívül függvény. Eldönti, hogy a kör kívül van-e egy adott sugarú, origo középpontú körön.
- [~Kor](#) () override

Publikus tagfüggvények a(z) **Sikidom** osztályból származnak

- **Sikidom** (const **Sikidom** &)
- **Sikidom** & **operator=** (const **Sikidom** &)
Sikidom osztály egyenlőség operátora.
- **Pont** **getkp** () const
- **Pont** **getp** () const
- bool **Tartalmazza** (const int) const
- **Sikidom** & **Mozgat** (const int _x, const int _y)
Egy sokszög mozgatását teszi lehetővé.
- **Sikidom** & **Forgat** (const double, const **Pont** &)
Egy sokszög forgatását teszi lehetővé.
- virtual ~**Sikidom** ()=0
Sikidom osztály destruktora. nincs dinamikusan foglalt adattag, tehát nem kell semmit felszabadítani.

Barátok

- std::ostream & **operator<<** (std::ostream &, const **Kor** &)

További örökölt tagok

Statikus publikus tagfüggvények a(z) **Sikidom** osztályból származnak

- static **Sikidom** * **createSikidom** (const std::string &type)
Sikidom osztály statikus factory függvénye. A függvény létrehoz egy új síkidomot a megadott típus alapján.

Védett tagfüggvények a(z) **Sikidom** osztályból származnak

- **Sikidom** (**Pont** _kp=**Pont**(0, 0), **Pont** _p=**Pont**(0, 0))
Sikidom konstruktor.

Védett attribútumok a(z) **Sikidom** osztályból származnak

- **Pont** **kp**
Középpont.
- **Pont** **p**
A síkidom egy csúcsa.

4.4.1. Részletes leírás

Kor osztály. A **Kor** osztály a **Sikidom** leszármazottja, mely egy kör síkidomot reprezentál, középpontjával és egy csúcsával.

4.4.2. Konstruktorkok és destruktorkok dokumentációja

4.4.2.1. Kor() [1/2]

```
Kor::Kor (
    Pont _kp = Pont (0, 0),
    Pont _p = Pont (0, 0) ) [inline]
```

4.4.2.2. Kor() [2/2]

```
Kor::Kor (
    const Kor & _k ) [inline]
```

4.4.2.3. ~Kor()

```
Kor::~Kor ( ) [inline], [override]
```

4.4.3. Tagfüggvények dokumentációja

4.4.3.1. Kivul()

```
bool Kor::Kivul (
    const std::size_t r ) const [override], [virtual]
```

[Kor](#) kívül függvény. Eldönti, hogy a kör kívül van-e egy adott sugarú, origo középpontú körön.

Paraméterek

<i>r</i>	a kör sugara.
----------	---------------

Megvalósítja a következőket: [Sikidom](#).

4.4.3.2. Rajtavan()

```
bool Kor::Rajtavan (
    const Pont & _p ) const [override], [virtual]
```

[Kor](#) Rajtavan függvény. A függvény eldönti, hogy egy pont rajta van-e a körön.

Paraméterek

\leftarrow	a vizsgált pont.
$_ \leftarrow$	
<i>p</i>	

Visszatérési érték

true, ha a pont rajta van a körön, egyébként false.

Megvalósítja a következőket: [Sikidom](#).

4.4.3.3. Read()

```
void Kor::Read (
    std::istream & is ) [override], [virtual]
```

[Kor](#) beolvasó függvény. A függvény beolvassa a kör adatait a megadott input streamből.

Paraméterek

<i>is</i>	a bemeneti stream.
-----------	--------------------

Megvalósítja a következőket: [Sikidom](#).

4.4.3.4. Terulet()

```
double Kor::Terulet ( ) const [override], [virtual]
```

[Kor](#) területét számoló függvény. A kiszamitas modja: $r^2 * \pi$.

Visszatérési érték

A kör területe.

Megvalósítja a következőket: [Sikidom](#).

4.4.3.5. Write()

```
void Kor::Write (
    std::ostream & os ) const [override], [virtual]
```

[Kor](#) kiíró függvény. A függvény kiírja a kör adatait a megadott output streambe.

Paraméterek

<i>os</i>	a kimeneti stream.
-----------	--------------------

Megvalósítja a következőket: [Sikidom](#).

4.4.4. Barát és kapcsolódó függvények dokumentációja

4.4.4.1. operator<<

```
std::ostream & operator<< (
    std::ostream & ,
    const Kor & ) [friend]
```

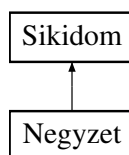
Ez a dokumentáció az osztályról a következő fájlok alapján készült:

- src/sikidom.h
- src/sikidom.cpp

4.5. Negyzet osztályreferencia

```
#include <sikidom.h>
```

A Negyzet osztály származási diagramja:



Publikus tagfüggvények

- **Negyzet** (**Pont** _kp=**Pont**(0, 0), **Pont** _p=**Pont**(0, 0))
- **Negyzet** (const **Negyzet** &_n)
- double **Terulet** () const override
*Negyzet területét számoló függvény. A kiszámítás módja: $R^2 * 2$.*
- bool **Rajtavan** (const **Pont** &) const override
Negyzet Rajtavan függvény. A függvény eldönti, hogy egy pont rajta van-e a négyzeten.
- void **Write** (std::ostream &os) const override
Negyzet kiíró függvény. A függvény kiírja a négyzet adatait a megadott output streambe.
- void **Read** (std::istream &is) override
Negyzet beolvasó függvény. A függvény beolvassa a négyzet adatait a megadott input streamből.
- bool **Kivul** (const std::size_t r) const override
Negyzet Kivul függvény. Eldönti, hogy a négyzet kívül van-e egy adott sugarú, origo középpontú körön.
- ~**Negyzet** () override

Publikus tagfüggvények a(z) **Sikidom** osztályból származnak

- **Sikidom** (const **Sikidom** &)
- **Sikidom** & **operator=** (const **Sikidom** &)
Sikidom osztály egyenlőség operátora.
- **Pont** **getkp** () const
- **Pont** **getp** () const
- bool **Tartalmazza** (const int) const
- **Sikidom** & **Mozgat** (const int _x, const int _y)
Egy sokszög mozgatását teszi lehetővé.
- **Sikidom** & **Forgat** (const double, const **Pont** &)
Egy sokszög forgatását teszi lehetővé.
- virtual ~**Sikidom** ()=0
Sikidom osztály destruktora. nincs dinamikusan foglalt adattag, tehát nem kell semmit felszabadítani.

Barátok

- `std::ostream & operator<< (std::ostream &, const Negyzet &)`

További örökölt tagok**Statikus publikus tagfüggvények a(z) Sikidom osztályból származnak**

- `static Sikidom * createSikidom (const std::string &type)`
Sikidom osztály statikus factory függvénye. A függvény létrehoz egy új síkidomot a megadott típus alapján.

Védett tagfüggvények a(z) Sikidom osztályból származnak

- `Sikidom (Pont _kp=Pont(0, 0), Pont _p=Pont(0, 0))`
Sikidom konstruktor.

Védett attribútumok a(z) Sikidom osztályból származnak

- `Pont kp`
Középpont.
- `Pont p`
A síkidom egy csúcsa.

4.5.1. Részletes leírás**Paraméterek**

Negyzet	osztály. A Negyzet osztály a Sikidom leszármazottja, mely egy négyzet síkidomot reprezentál, középpontjával és egy csúcsával.
----------------	---

4.5.2. Konstruktorok és destruktorok dokumentációja**4.5.2.1. Negyzet() [1/2]**

```
Negyzet::Negyzet (
    Pont _kp = Pont (0, 0),
    Pont _p = Pont (0, 0) ) [inline]
```

4.5.2.2. Negyzet() [2/2]

```
Negyzet::Negyzet (
    const Negyzet & _n ) [inline]
```

4.5.2.3. ~Negyzet()

```
Negyzet::~Negyzet ( ) [inline], [override]
```

4.5.3. Tagfüggvények dokumentációja

4.5.3.1. Kivul()

```
bool Negyzet::Kivul (
    const std::size_t r ) const [override], [virtual]
```

[Negyzet](#) Kivul függvény. Eldönti, hogy a négyzet kívül van-e egy adott sugarú, origo középpontú körön.

Paraméterek

<i>r</i>	a kör sugara.
----------	---------------

Visszatérési érték

true, ha a négyzet kívül van a körön, egyébként false.

Megvalósítja a következőket: [Sikidom](#).

4.5.3.2. Rajtavan()

```
bool Negyzet::Rajtavan (
    const Pont & P ) const [override], [virtual]
```

[Negyzet](#) Rajtavan függvény. A függvény eldönti, hogy egy pont rajta van-e a négyzeten.

Paraméterek

<i>P</i>	a vizsgált pont.
----------	------------------

Visszatérési érték

true, ha a pont rajta van a négyzeten, egyébként false.

Megvalósítja a következőket: [Sikidom](#).

4.5.3.3. Read()

```
void Negyzet::Read (
    std::istream & is ) [override], [virtual]
```

[Negyzet](#) beolvasó függvény. A függvény beolvassa a négyzet adatait a megadott input streamből.

Paraméterek

<i>is</i>	a bemeneti stream.
-----------	--------------------

Megvalósítja a következőket: [Sikidom](#).

4.5.3.4. Terulet()

```
double Negyzet::Terulet ( ) const [override], [virtual]
```

[Negyzet](#) területét számoló függvény. A kiszamitas modja: $R^2 * 2$.

Visszatérési érték

A négyzet területe.

Megvalósítja a következőket: [Sikidom](#).

4.5.3.5. Write()

```
void Negyzet::Write (
    std::ostream & os ) const [override], [virtual]
```

[Negyzet](#) kiíró függvény. A függvény kiírja a négyzet adatait a megadott output streambe.

Paraméterek

<code>os</code>	a kimeneti stream.
-----------------	--------------------

Megvalósítja a következőket: [Sikidom](#).

4.5.4. Barát és kapcsolódó függvények dokumentációja

4.5.4.1. operator<<

```
std::ostream & operator<< (
    std::ostream & ,
    const Negyzet & ) [friend]
```

Ez a dokumentáció az osztályról a következő fájlok alapján készült:

- [src/sikidom.h](#)
- [src/sikidom.cpp](#)

4.6. Pont osztályreferencia

[Pont](#) osztály A pontokat a síkon tárolja, x és y koordinátákkal.

```
#include <pont.h>
```

Publikus tagfüggvények

- `Pont` (`const double _x=0, const double _y=0`)
`Pont` osztály másoló konstruktora.
- `Pont` (`const Pont &`)
`Pont` osztály értékadó operátora.
- `Pont & operator=` (`const Pont &`)
`Pont` osztály `x` koordinátájának beállítása.
- `Pont & setx` (`double`)
`Pont` osztály `y` koordinátájának beállítása.
- `Pont & sety` (`double`)
`Pont` osztály `y` koordinátájának beállítása.
- `double getx` (`() const`)
`Pont` osztály `x` koordinátájának lekérdezése.
- `double gety` (`() const`)
`Pont` osztály `y` koordinátájának lekérdezése.
- `Pont operator+` (`const Pont &`) `const`
`Pont` osztály összeadó operátora.
- `Pont operator-` (`const Pont &`) `const`
`Pont` osztály kivonó operátora.
- `bool operator==` (`const Pont &`) `const`
`Pont` osztály egyenlőség operátora.
- `bool operator!=` (`const Pont &`) `const`
`Pont` osztály egyenlőtlenség operátora.
- `double dst` (`const Pont &`) `const`
`Pont` osztály távolság számító metódusa.
- `Pont & Mozgat` (`const double _x=0, const double _y=0`)
`Pont` osztály mozgató metódusa.
- `Pont & Forgat` (`const double, const Pont &`)
`Pont` osztály forgató metódusa.

Privát attribútumok

- `double x`
`Pont` `x` koordinátája.
- `double y`
`Pont` `y` koordinátája.

Barátok

- `std::ostream & operator<<` (`std::ostream &, const Pont &`)
`Pont` osztály kiíró operátora.
- `std::istream & operator>>` (`std::istream &, Pont &`)
`Pont` osztály beolvasó operátora. formátumok: "(x,y)" vagy "x,y".

4.6.1. Részletes leírás

`Pont` osztály A pontokat a síkon tárolja, `x` és `y` koordinátákkal.

4.6.2. Konstruktorkok és destruktorkok dokumentációja

4.6.2.1. Pont() [1/2]

```
Pont::Pont (
    const double _x = 0,
    const double _y = 0 ) [inline]
```

4.6.2.2. Pont() [2/2]

```
Pont::Pont (
    const Pont & p )
```

Pont osztály másoló konstruktora.

Paraméterek

<i>p</i>	másolandó pont
----------	----------------

4.6.3. Tagfüggvények dokumentációja

4.6.3.1. dst()

```
double Pont::dst (
    const Pont & p ) const
```

Pont osztály távolság számító metódusa.

Paraméterek

<i>p</i>	a másik pont
----------	--------------

Visszatérési érték

a két pont távolsága

4.6.3.2. Forgat()

```
Pont & Pont::Forgat (
    const double rad,
    const Pont & center = Pont(0, 0) )
```

Pont osztály forgató metódusa.

Paraméterek

<i>rad</i>	forgatás szöge radiánban
<i>center</i>	a forgatás középpontja

Visszatérési érték

*this

4.6.3.3. getx()

```
double Pont::getx ( ) const
```

Pont osztály x koordinátájának lekérdezése.

Visszatérési érték

x koordináta értéke

4.6.3.4. gety()

```
double Pont::gety ( ) const
```

Pont osztály y koordinátájának lekérdezése.

Visszatérési érték

y koordináta értéke

4.6.3.5. Mozgat()

```
Pont & Pont::Mozgat (
    const double _x = 0,
    const double _y = 0 )
```

Pont osztály mozgató metódusa.

Paraméterek

↔ _↔ x	x irányú elmozdulás
↔ _↔ y	y irányú elmozdulás

Visszatérési érték

*this

4.6.3.6. operator"!="()

```
bool Pont::operator!= (
    const Pont & p ) const
```


`Pont` osztály egyenlőtlenség operátora.

Paraméterek

p	a másik pont
-----	--------------

Visszatérési érték

igaz, ha a két pont koordinátái nem egyenlők, egyébként hamis.

4.6.3.7. `operator+()`

```
Pont Pont::operator+ (
    const Pont & b ) const
```

`Pont` osztály összeadó operátora.

Paraméterek

b	a másik pont
-----	--------------

Visszatérési érték

az összeg

4.6.3.8. `operator-()`

```
Pont Pont::operator- (
    const Pont & b ) const
```

`Pont` osztály kivonó operátora.

Paraméterek

b	a másik pont
-----	--------------

Visszatérési érték

a különbség

4.6.3.9. `operator=()`

```
Pont & Pont::operator= (
    const Pont & p )
```

`Pont` osztály értékadó operátora.

Paraméterek

p	másolandó pont
-----	----------------

Visszatérési érték

*this

4.6.3.10. operator==()

```
bool Pont::operator== (
    const Pont & p ) const
```

Pont osztály egyenlőség operátora.

Paraméterek

p	a másik pont
-----	--------------

Visszatérési érték

igaz, ha a két pont koordinátái egyenlők, egyébként hamis.

4.6.3.11. setx()

```
Pont & Pont::setx (
    double _x )
```

Pont osztály x koordinátájának beállítása.

Paraméterek

↔	beállítandó x koordináta
↔	
x	

Visszatérési érték

*this

4.6.3.12. sety()

```
Pont & Pont::sety (
    double _y )
```

Pont osztály y koordinátájának beállítása.

Paraméterek

\leftrightarrow	beállítandó y koordináta
$_ \leftrightarrow$	
y	

Visszatérési érték

*this

4.6.4. Barát és kapcsolódó függvények dokumentációja

4.6.4.1. operator<<

```
std::ostream & operator<< (
    std::ostream & os,
    const Pont & p ) [friend]
```

Pont osztály kiíró operátora.

Paraméterek

<i>os</i>	output stream referencia
<i>p</i>	a kiírandó pont

Visszatérési érték

output stream referencia

4.6.4.2. operator>>

```
std::istream & operator>> (
    std::istream & is,
    Pont & p ) [friend]
```

Pont osztály beolvasó operátora. formátumok: "(x,y)" vagy "x,y".

Paraméterek

<i>is</i>	input stream referencia
<i>p</i>	a beolvasandó pont

Visszatérési érték

input stream referencia

4.6.5. Adattagok dokumentációja**4.6.5.1. x**

```
double Pont::x [private]
```

Pont x koordinátája.

4.6.5.2. y

```
double Pont::y [private]
```

Pont y koordinátája.

Ez a dokumentáció az osztályról a következő fájlok alapján készült:

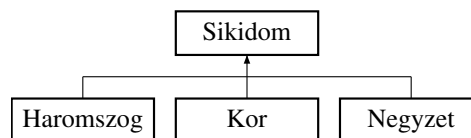
- [src/pont.h](#)
- [src/pont.cpp](#)

4.7. Sikidom osztályreferencia

Sikidom osztály. A sikidom osztály egy absztrakt osztály, melynek leszármazottai a különböző (szabályos) síkidomokat reprezentálják, középpontjukkal és egy csúcsukkal.

```
#include <sikidom.h>
```

A Sikidom osztály származási diagramja:

**Publikus tagfüggvények**

- **Sikidom** (const **Sikidom** &)
- **Sikidom** & **operator=** (const **Sikidom** &)
Sikidom osztály egyenlőség operátora.
- **Pont** **getkp** () const
- **Pont** **getp** () const
- bool **Tartalmazza** (const int) const
- **Sikidom** & **Mozgat** (const int _x, const int _y)
Egy sokszög mozgatását teszi lehetővé.
- **Sikidom** & **Forgat** (const double, const **Pont** &)
Egy sokszög forgatását teszi lehetővé.
- virtual double **Terulet** () const =0
A sokszög területét adja vissza.
- virtual bool **Rajtavan** (const **Pont** &) const =0
- virtual void **Write** (std::ostream &os) const =0
- virtual void **Read** (std::istream &is)=0
- virtual bool **Kivul** (const std::size_t r) const =0
- virtual **~Sikidom** ()=0

Sikidom osztály destruktora. nincs dinamikusan foglalt adattag, tehát nem kell semmit felszabadítani.

Statikus publikus tagfüggvények

- static `Sikidom * createSikidom (const std::string &type)`
Sikidom osztály statikus factory függvénye. A függvény létrehoz egy új síkidomot a megadott típus alapján.

Védett tagfüggvények

- `Sikidom (Pont _kp=Pont(0, 0), Pont _p=Pont(0, 0))`
Sikidom konstruktor.

Védett attribútumok

- `Pont kp`
Középpont.
- `Pont p`
A síkidom egy csúcsa.

Barátok

- `std::ostream & operator<< (std::ostream &, const Sikidom *const)`
Sikidom osztály kiíró operátora. A függvény kiírja a síkidomot a megadott output streambe.
- `std::istream & operator>> (std::istream &is, Sikidom **sikidom)`
Sikidom osztály beolvasó operátora. A függvény beolvassa a síkidomot a megadott input streamből.

4.7.1. Részletes leírás

`Sikidom` osztály. A síkidom osztály egy absztrakt osztály, melynek leszármazottai a különböző (szabályos) síkidomokat reprezentálják, középpontjukkal és egy csúcsukkal.

4.7.2. Konstruktorok és destruktorok dokumentációja

4.7.2.1. Sikidom() [1/2]

```
Sikidom::Sikidom (
    Pont _kp = Pont (0, 0),
    Pont _p = Pont (0, 0) ) [inline], [protected]
```

`Sikidom` konstruktor.

Paraméterek

<code>_kp</code>	középpont.
<code>_p</code>	egy csúcs.

4.7.2.2. Sikidom() [2/2]

```
Sikidom::Sikidom (
    const Sikidom & s )
```

Sikidom osztály Copy konstruktora.

Paraméterek

s	Referencia a lemasolando Sikidomra
---	------------------------------------

4.7.2.3. ~Sikidom()

```
Sikidom::~~Sikidom ( ) [pure virtual]
```

Sikidom osztály destruktora. nincs dinamikusan foglalt adattag, tehát nem kell semmit felszabadítani.

4.7.3. Tagfüggvények dokumentációja

4.7.3.1. createSikidom()

```
Sikidom * Sikidom::createSikidom (
    const std::string & s ) [static]
```

Sikidom osztály statikus factory függvénye. A függvény létrehoz egy új síkidomot a megadott típus alapján.

Paraméterek

s	a létrehozandó síkidom típusa szöveg formátumban.
---	---

Visszatérési érték

A létrehozott síkidomra mutató pointer.

4.7.3.2. Forgat()

```
Sikidom & Sikidom::Forgat (
    const double rad,
    const Pont & center = Pont (0, 0) )
```

Egy sokszög forgatását teszi lehetővé.

Sikidom osztály forgató függvénye. A függvény elforgatja a síkidomot <rad> radiánnal.

Paraméterek

deg	forgatás mértéke, fokban.
rad	forgatás mértéke, radiánban.
center	a forgatás középpontja.

Visszatérési érték

Referencia a forgatott síkidomra.

4.7.3.3. getkp()

```
Pont Sikidom::getkp ( ) const [inline]
```

4.7.3.4. getp()

```
Pont Sikidom::getp ( ) const [inline]
```

4.7.3.5. Kivul()

```
virtual bool Sikidom::Kivul (
    const std::size_t r ) const [pure virtual]
```

Megvalósítják a következők: [Kor](#), [Haromszog](#) és [Negyzet](#).

4.7.3.6. Mozgat()

```
Sikidom & Sikidom::Mozgat (
    const int _x,
    const int _y )
```

Egy sokszög mozgatását teszi lehetővé.

[Sikidom](#) osztály mozgató függvénye. A függvény elmozgatja a síkidomot az x és y értékekkel.

Paraméterek

↵ _↵ x	x tengely irányú mozgatás mértéke.
↵ _↵ y	y tengely irányú mozgatás mértéke.
↵ _↵ x	x tengely irányú mozgatás mértéke.
↵ _↵ y	y tengely irányú mozgatás mértéke.

Visszatérési érték

Referencia a mozgatott síkidomra.

4.7.3.7. operator=()

```
Sikidom & Sikidom::operator= (
    const Sikidom & s )
```

[Sikidom](#) osztály egyenlőség operatora.

Paraméterek

s	
---	--

Visszatérési érték

Az egymásután fűzhető műveletek miatt [Sikidom](#) referencia.

4.7.3.8. Rajtavan()

```
virtual bool Sikidom::Rajtavan (
    const Pont & ) const [pure virtual]
```

Megvalósítják a következők: [Kor](#), [Haromszog](#) és [Negyzet](#).

4.7.3.9. Read()

```
virtual void Sikidom::Read (
    std::istream & is ) [pure virtual]
```

Megvalósítják a következők: [Kor](#), [Haromszog](#) és [Negyzet](#).

4.7.3.10. Tartalmazza()

```
bool Sikidom::Tartalmazza (
    const int ) const
```

4.7.3.11. Terulet()

```
virtual double Sikidom::Terulet ( ) const [pure virtual]
```

A sokszög területét adja vissza.

Megvalósítják a következők: [Kor](#), [Haromszog](#) és [Negyzet](#).

4.7.3.12. Write()

```
virtual void Sikidom::Write (
    std::ostream & os ) const [pure virtual]
```

Megvalósítják a következők: [Kor](#), [Haromszog](#) és [Negyzet](#).

4.7.4. Barát és kapcsolódó függvények dokumentációja

4.7.4.1. operator<<

```
std::ostream & operator<< (  
    std::ostream & os,  
    const Sikidom * const sikidom ) [friend]
```

[Sikidom](#) osztály kiíró operátora. A függvény kiírja a síkidomot a megadott output streambe.

Paraméterek

<i>os</i>	a kimeneti stream.
<i>sikidom</i>	a kiírandó síkidomra mutató pointer.

Visszatérési érték

A kimeneti stream referenciája.

4.7.4.2. operator>>

```
std::istream & operator>> (
    std::istream & is,
    Sikidom ** sikidom ) [friend]
```

[Sikidom](#) osztály beolvasó operátora. A függvény beolvassa a síkidomot a megadott input streamből.

Paraméterek

<i>is</i>	a bemeneti stream.
<i>sikidom</i>	a beolvasott síkidomra mutató pointer.

Visszatérési érték

A bemeneti stream referenciája.

4.7.5. Adattagok dokumentációja

4.7.5.1. kp

```
Pont Sikidom::kp [protected]
```

Középpont.

4.7.5.2. p

```
Pont Sikidom::p [protected]
```

A síkidom egy csúcsa.

Ez a dokumentáció az osztályról a következő fájlok alapján készült:

- [src/sikidom.h](#)
- [src/sikidom.cpp](#)

4.8. Tomb< T > osztálysablon-referencia

Generikus Dinamikus tömb osztály. A `Tomb` osztály egy dinamikus tömböt reprezentál, melynek mérete a felhasználó által nem korlátozott.

```
#include <tomb.hpp>
```

Osztályok

- class `const_iterator`
`const_iterator` osztály. Az `const_iterator` osztály a `Tomb` osztály `const_iterator`-a.
- class `iterator`
`iterator` osztály. Az `iterator` osztály a `Tomb` osztály `iterator`-a.

Publikus tagfüggvények

- `Tomb ()`
`Tomb` osztály konstruktora. A konstruktor létrehoz egy üres tömböt, 1 kapacitással.
- `iterator begin ()`
`begin` iterator.
- `iterator end ()`
`end` iterator.
- `const_iterator begin () const`
`begin` `const_iterator`.
- `const_iterator end () const`
`end` `const_iterator`.
- `size_t size () const`
`Tomb` méretét visszaado függvény.
- `size_t capacity () const`
`Tomb` kapacitását visszaado függvény.
- `T & at (const size_t i)`
`Tomb` `at` függvénye.
- `T & operator[] (const size_t i)`
`Tomb` `operator[]` függvénye.
- `const T & at (const size_t i) const`
`Tomb` `const at` függvénye.
- `const T & operator[] (const size_t i) const`
`Tomb` `const operator[]` függvénye.
- `void resize (size_t newSize)`
`Tomb` `resize` függvénye. A függvény a tömb méretét növeli a megadott értékkel.
- `void push_back (T value)`
`Tomb` `push_back` függvénye. A függvény a tömb végére fűz egy elemet.
- `int find (const T &value) const`
`Tomb` `find` függvénye. A függvény megkeresi az első olyan elemet, mely megegyezik az értékkel.
- `virtual ~Tomb ()`
`Tomb` destruktora. A destruktork felszabadítja a tömböt.

Privát attribútumok

- `T * t`
A dinamikusan foglalt tömb elejére mutató pointer.
- `size_t currentSize`
A tömb aktuális mérete.
- `size_t currentCapacity`
A tömb aktuális kapacitása.

4.8.1. Részletes leírás

```
template<class T>
class Tomb< T >
```

Generikus Dinamikus tömb osztály. A `Tomb` osztály egy dinamikus tömböt reprezentál, melynek mérete a felhasználó által nem korlátozott.

4.8.2. Konstruktorkok és destruktorok dokumentációja

4.8.2.1. `Tomb()`

```
template<class T >
Tomb< T >::Tomb ( ) [inline]
```

`Tomb` osztály konstruktora. A konstruktor létrehoz egy üres tömböt, 1 kapacitással.

4.8.2.2. `~Tomb()`

```
template<class T >
virtual Tomb< T >::~~Tomb ( ) [inline], [virtual]
```

`Tomb` destruktor. A destruktor felszabadítja a tömböt.

4.8.3. Tagfüggvények dokumentációja

4.8.3.1. `at()` [1/2]

```
template<class T >
T & Tomb< T >::at (
    const size_t i ) [inline]
```

`Tomb` `at` függvénye.

Paraméterek

<i>i</i>	Az index, ahol lévő az elemet szeretnénk elérni.
----------	--

Visszatérési érték

Az i-edik elem.

4.8.3.2. at() [2/2]

```
template<class T >
const T & Tomb< T >::at (
    const size_t i ) const [inline]
```

Tomb const at függvénye.

Paraméterek

<i>i</i>	Az index, ahol lévő elemet szeretnénk elérni.
----------	---

Visszatérési érték

const referencia az i-edik elemre.

4.8.3.3. begin() [1/2]

```
template<class T >
iterator Tomb< T >::begin ( ) [inline]
```

begin iterator.

Visszatérési érték

Az első elemre mutató iterator.

4.8.3.4. begin() [2/2]

```
template<class T >
const_iterator Tomb< T >::begin ( ) const [inline]
```

begin const_iterator.

Visszatérési érték

Az első elemre mutató const_iterator.

4.8.3.5. capacity()

```
template<class T >
size_t Tomb< T >::capacity ( ) const [inline]
```

Tomb kapacitását visszaado függvény.

Visszatérési érték

A tömb kapacitása.

4.8.3.6. end() [1/2]

```
template<class T >
iterator Tomb< T >::end ( ) [inline]
```

end iterator.

Visszatérési érték

Az utolsó elem után mutató iterator.

4.8.3.7. end() [2/2]

```
template<class T >
const_iterator Tomb< T >::end ( ) const [inline]
```

end `const_iterator`.

Visszatérési érték

Az utolsó elem után mutató `const_iterator`.

4.8.3.8. find()

```
template<class T >
int Tomb< T >::find (
    const T & value ) const [inline]
```

`Tomb` find függvénye. A függvény megkeresi az első olyan elemet, mely megegyezik az értékkel.

Paraméterek

<code>value</code>	Az érték, melyet keresünk.
--------------------	----------------------------

Visszatérési érték

Az első olyan elem indexe, mely megegyezik az értékkel, ha nincs ilyen, akkor -1.

4.8.3.9. operator[]() [1/2]

```
template<class T >
T & Tomb< T >::operator[] (
    const size_t i ) [inline]
```

`Tomb` `operator[]` függvénye.

Paraméterek

<i>i</i>	Az index, ahol lévő elemet szeretnénk elérni.
----------	---

Visszatérési érték

Az i-edik elem.

4.8.3.10. operator[]() [2/2]

```
template<class T >
const T & Tomb< T >::operator[] (
    const size_t i ) const [inline]
```

[Tomb](#) const operator[] függvénye.

Paraméterek

<i>i</i>	Az index, ahol lévő elemet szeretnénk elérni.
----------	---

Visszatérési érték

const referencia az i-edik elemre.

4.8.3.11. push_back()

```
template<class T >
void Tomb< T >::push_back (
    T value ) [inline]
```

[Tomb](#) push_back függvénye. A függvény a tömb végére fűz egy elemet.

Paraméterek

<i>value</i>	Az érték, melyet a tömb végére fűzünk.
--------------	--

4.8.3.12. resize()

```
template<class T >
void Tomb< T >::resize (
    size_t newSize ) [inline]
```

[Tomb](#) resize függvénye. A függvény a tömb méretét növeli a megadott értékkel.

Paraméterek

<i>newSize</i>	A novelendo meret.
----------------	--------------------

4.8.3.13. size()

```
template<class T >
size_t Tomb< T >::size ( ) const [inline]
```

[Tomb](#) méretét visszaado függvény.

Visszatérési érték

A tömb mérete.

4.8.4. Adattagok dokumentációja

4.8.4.1. currentCapacity

```
template<class T >
size_t Tomb< T >::currentCapacity [private]
```

A tömb aktuális kapacitása.

4.8.4.2. currentSize

```
template<class T >
size_t Tomb< T >::currentSize [private]
```

A tömb aktuális mérete.

4.8.4.3. t

```
template<class T >
T* Tomb< T >::t [private]
```

A dinamikusan foglalt tömb elejére mutató pointer.

Ez a dokumentáció az osztályról a következő fájl alapján készült:

- [src/tomb.hpp](#)

5. fejezet

Fájlok dokumentációja

5.1. src/pont.cpp fájlreferencia

```
#include "pont.h"
#include <cmath>
#include <stdexcept>
```

Függvények

- `std::ostream & operator<< (std::ostream &os, const Pont &p)`
Pont osztály kiíró operátora.
- `std::istream & operator>> (std::istream &is, Pont &p)`
Pont osztály beolvasó operátora. formátumok: "(x,y)" vagy "x,y".
- `double dst (const Pont &a, const Pont &b)`
két pont közötti távolság számító függvény.

5.1.1. Függvények dokumentációja

5.1.1.1. dst()

```
double dst (
    const Pont & a,
    const Pont & b )
```

két pont közötti távolság számító függvény.

Paraméterek

<i>a</i>	egyik pont
<i>b</i>	másik pont

Visszatérési érték

a két pont távolsága

5.1.1.2. operator<<()

```
std::ostream & operator<< (
    std::ostream & os,
    const Pont & p )
```

Pont osztály kiíró operátora.

Paraméterek

<i>os</i>	output stream referencia
<i>p</i>	a kiírandó pont

Visszatérési érték

output stream referencia

5.1.1.3. operator>>()

```
std::istream & operator>> (
    std::istream & is,
    Pont & p )
```

Pont osztály beolvasó operátora. formátumok: "(x,y)" vagy "x,y".

Paraméterek

<i>is</i>	input stream referencia
<i>p</i>	a beolvasandó pont

Visszatérési érték

input stream referencia

5.2. src/pont.h fájlreferencia

```
#include <iostream>
```

Osztályok

- class **Pont**
Pont osztály A pontokat a síkon tárolja, x és y koordinátákkal.

Függvények

- double `dst` (const `Pont` &a, const `Pont` &b)
két pont közötti távolság számító függvény.

5.2.1. Függvények dokumentációja

5.2.1.1. `dst()`

```
double dst (
    const Pont & a,
    const Pont & b )
```

két pont közötti távolság számító függvény.

Paraméterek

<i>a</i>	egyik pont
<i>b</i>	másik pont

Visszatérési érték

a két pont távolsága

5.3. pont.h

[Ugrás a fájl dokumentációjához.](#)

```
00001 #include <iostream>
00002
00003 #ifndef SIKIDOM_PONT_H
00004 #define SIKIDOM_PONT_H
00005
00010 class Pont{
00011 private:
00015     double x;
00016
00020     double y;
00021 public:
00022     Pont(const double _x = 0, const double _y = 0): x(_x), y(_y){}
00023
00024     Pont(const Pont&);
00025
00026     Pont& operator=(const Pont&);
00027
00028     Pont& setx(double);
00029
00030     Pont& sety(double);
00031
00032     double getx() const;
00033
00034     double gety() const;
00035
00036     Pont operator+(const Pont&) const;
00037
00038     Pont operator-(const Pont&) const;
00039
00040     bool operator==(const Pont&) const;
00041
00042     bool operator!=(const Pont&) const;
00043
00044     double dst(const Pont&) const;
00045
00046     Pont& Mozgat(const double _x = 0, const double _y = 0);
```

```

00047
00048     Pont& Forgat(const double, const Pont&);
00049
00050     friend std::ostream& operator<<(std::ostream&, const Pont&);
00051
00052     friend std::istream& operator>>(std::istream&, Pont&);
00053 };
00054
00055 double dst(const Pont& a, const Pont& b);
00056
00057 #endif //SIKIDOM_PONT_H

```

5.4. src/sikidom.cpp fájlreferencia

```

#include <cmath>
#include <istream>
#include <algorithm>
#include "pont.h"
#include "sikidom.h"

```

Függvények

- std::istream & operator>> (std::istream &is, Sikidom **sikidom)
Sikidom osztály beolvasó operátora. A függvény beolvassa a síkidomot a megadott input streamből.
- std::ostream & operator<< (std::ostream &os, const Sikidom *const sikidom)
Sikidom osztály kiíró operátora. A függvény kiírja a síkidomot a megadott output streambe.
- bool IsOnTriangle (const Pont &P, const Pont &A, const Pont &B, const Pont &C)
IsOnTriangle függvény. A függvény eldönti, hogy egy pont rajta van-e egy háromszögön.

5.4.1. Függvények dokumentációja

5.4.1.1. IsOnTriangle()

```

bool IsOnTriangle (
    const Pont & P,
    const Pont & A,
    const Pont & B,
    const Pont & C )

```

IsOnTriangle függvény. A függvény eldönti, hogy egy pont rajta van-e egy háromszögön.

Paraméterek

<i>P</i>	a vizsgált pont.
<i>A</i>	háromszög egyik csúcsa.
<i>B</i>	háromszög másik csúcsa.
<i>C</i>	háromszög harmadik csúcsa.

Visszatérési érték

true, ha a pont rajta van a háromszögön, egyébként false.

A kiszamitas modja: https://en.wikipedia.org/wiki/Barycentric_coordinate_system#Determining_location_with_respect_to_a_triangle

5.4.1.2. operator<<()

```
std::ostream & operator<< (
    std::ostream & os,
    const Sikidom *const sikidom )
```

Sikidom osztály kiíró operátora. A függvény kiírja a síkidomot a megadott output streambe.

Paraméterek

<i>os</i>	a kimeneti stream.
<i>sikidom</i>	a kiírandó síkidomra mutató pointer.

Visszatérési érték

A kimeneti stream referenciája.

5.4.1.3. operator>>()

```
std::istream & operator>> (
    std::istream & is,
    Sikidom ** sikidom )
```

Sikidom osztály beolvasó operátora. A függvény beolvassa a síkidomot a megadott input streamből.

Paraméterek

<i>is</i>	a bemeneti stream.
<i>sikidom</i>	a beolvasott síkidomra mutató pointer.

Visszatérési érték

A bemeneti stream referenciája.

5.5. src/sikidom.h fájlreferencia

```
#include "pont.h"
#include <cstdint>
#include <iostream>
#include <memory>
```

Osztályok

- class [Sikidom](#)
Sikidom osztály. A sikidom osztály egy absztrakt osztály, melynek leszármazottai a különböző (szabályos) síkidomokat reprezentálják, középpontjukkal és egy csúcsukkal.
- class [Kor](#)
Kor osztály. A Kor osztály a Sikidom leszármazottja, mely egy kör síkidomot reprezentál, középpontjával és egy csúcsával.
- class [Haromszog](#)
Haromszog osztály. A Haromszog osztály a Sikidom leszármazottja, mely egy háromszög síkidomot reprezentál, középpontjával és egy csúcsával.
- class [Negyzet](#)

Függvények

- bool [IsOnTriangle](#) (const [Pont](#) &, const [Pont](#) &, const [Pont](#) &, const [Pont](#) &)
IsOnTriangle függvény. A függvény eldönti, hogy egy pont rajta van-e egy háromszögön.

5.5.1. Függvények dokumentációja

5.5.1.1. IsOnTriangle()

```
bool IsOnTriangle (
    const Pont & P,
    const Pont & A,
    const Pont & B,
    const Pont & C )
```

IsOnTriangle függvény. A függvény eldönti, hogy egy pont rajta van-e egy háromszögön.

Paraméterek

<i>P</i>	a vizsgált pont.
<i>A</i>	háromszög egyik csúcsa.
<i>B</i>	háromszög másik csúcsa.
<i>C</i>	háromszög harmadik csúcsa.

Visszatérési érték

true, ha a pont rajta van a háromszögön, egyébként false.

A kiszamitas modja: https://en.wikipedia.org/wiki/Barycentric_coordinate_system#Determining_location_with_respect_to_a_triangle

5.6. sikidom.h

[Ugrás a fájl dokumentációjához.](#)

```
00001 #include "pont.h"
```

```

00002 #include <cstdint>
00003 #include <iostream>
00004 #include <memory>
00005
00006 #ifndef SIKIDOM_SIKIDOM_H
00007 #define SIKIDOM_SIKIDOM_H
00008
00009
00015 class Sikidom {
00016 protected:
00019     Pont kp;
00020
00023     Pont p;
00024
00029     Sikidom(Pont _kp = Pont(0, 0), Pont _p = Pont(0, 0)) : kp(_kp), p(_p) {}
00030
00031 public:
00032     Sikidom(const Sikidom &);
00033
00034     Sikidom &operator=(const Sikidom &);
00035
00036     Pont getkp() const { return kp; }
00037
00038     Pont getp() const { return p; }
00039
00040     bool Tartalmazza(const int) const;
00041
00046     Sikidom &Mozgat(const int _x, const int _y);
00047
00051     Sikidom &Forgat(const double, const Pont &);
00052
00055     virtual double Terulet() const = 0;
00056
00057     virtual bool Rajtavan(const Pont &) const = 0;
00058
00059     virtual void Write(std::ostream& os) const = 0;
00060
00061     virtual void Read(std::istream& is) = 0;
00062
00063     static Sikidom* createSikidom(const std::string& type);
00064
00065     virtual bool Kivul(const std::size_t r) const = 0;
00066
00067     friend std::ostream &operator<<(std::ostream&, const Sikidom * const);
00068
00069     friend std::istream& operator>>(std::istream& is, Sikidom** sikidom);
00070
00071     virtual ~Sikidom() = 0;
00072 };
00073
00079 class Kor : public Sikidom {
00080 public:
00081     Kor(Pont _kp = Pont(0, 0), Pont _p = Pont(0, 0)) : Sikidom(_kp, _p) {}
00082
00083     Kor(const Kor &_k) : Sikidom(_k) {}
00084
00085     double Terulet() const override;
00086
00087     bool Rajtavan(const Pont &) const override;
00088
00089     void Write(std::ostream& os) const override;
00090
00091     void Read(std::istream& is) override;
00092
00093     bool Kivul(const std::size_t r) const override;
00094
00095     friend std::ostream &operator<<(std::ostream&, const Kor &);
00096
00097     ~Kor() override {}
00098 };
00099
00105 class Haromszog : public Sikidom {
00106 public:
00107     Haromszog(Pont _kp = Pont(0, 0), Pont _p = Pont(0, 0)) : Sikidom(_kp, _p) {}
00108
00109     Haromszog(const Haromszog &_h) : Sikidom(_h) {}
00110
00111     double Terulet() const override;
00112
00113     bool Rajtavan(const Pont &) const override;
00114
00115     void Write(std::ostream& os) const override;
00116
00117     void Read(std::istream& is) override;
00118
00119     bool Kivul(const std::size_t r) const override;
00120

```

```

00121     friend std::ostream &operator<<(std::ostream &, const Haromszog &);
00122
00123     ~Haromszog() override {}
00124 };
00125
00131 class Negyzet : public Sikidom {
00132 public:
00133     Negyzet(Pont _kp = Pont(0, 0), Pont _p = Pont(0, 0)) : Sikidom(_kp, _p) {}
00134
00135     Negyzet(const Negyzet &n) : Sikidom(_n) {}
00136
00137     double Terulet() const override;
00138
00139     bool Rajtavan(const Pont &) const override;
00140
00141     void Write(std::ostream& os) const override;
00142
00143     void Read(std::istream& is) override;
00144
00145     bool Kivul(const std::size_t r) const override;
00146
00147     friend std::ostream &operator<<(std::ostream &, const Negyzet &);
00148
00149     ~Negyzet() override {}
00150 };
00151
00152 bool IsOnTriangle(const Pont&, const Pont&, const Pont&, const Pont&);
00153
00154 #endif // !SIKIDOM_SIKIDOM_H

```

5.7. src/tomb.hpp fájlreferencia

```
#include <stdexcept>
```

Osztályok

- class `Tomb< T >`
Generikus Dinamikus tömb osztály. A `Tomb` osztály egy dinamikus tömböt reprezentál, melynek mérete a felhasználó által nem korlátozott.
- class `Tomb< T >::iterator`
iterator osztály. Az iterator osztály a `Tomb` osztály iteratora.
- class `Tomb< T >::const_iterator`
const_iterator osztály. Az const_iterator osztály a `Tomb` osztály const_iteratora.

5.8. tomb.hpp

[Ugrás a fájl dokumentációjához.](#)

```

00001 #ifndef MYTOMB
00002 #define MYTOMB
00003
00004 #include <stdexcept>
00005
00010 template <class T> class Tomb {
00013     T *t;
00016     size_t currentSize;
00019     size_t currentCapacity;
00020
00021 public:
00022     class iterator;
00023     class const_iterator;
00024
00028     Tomb() : t(new T[1]), currentSize(0), currentCapacity(1) {}
00029
00034     iterator begin() { return iterator(t, 0); }
00035
00040     iterator end() { return iterator(t, currentSize); }

```



```

00041
00046     const_iterator begin() const { return const_iterator(t, 0); }
00047
00052     const_iterator end() const { return const_iterator(t, currentSize); }
00053
00057     size_t size() const { return currentSize; }
00058
00062     size_t capacity() const { return currentCapacity; }
00063
00068     T &at(const size_t i) {
00069         if (i >= currentSize)
00070             throw std::out_of_range("Array.at(): invalid index");
00071         return t[i];
00072     }
00073
00078     T &operator[](const size_t i) { return at(i); }
00079
00084     const T &at(const size_t i) const {
00085         if (i >= currentSize)
00086             throw std::out_of_range("Array.at(): invalid index");
00087         return t[i];
00088     }
00089
00094     const T &operator[](const size_t i) const { return at(i); }
00095
00100     void resize(size_t newSize) {
00101         if (newSize > currentCapacity) {
00102             T* newT = new T[newSize * 2];
00103             for (size_t i = 0; i < currentSize; ++i) {
00104                 newT[i] = t[i];
00105             }
00106             delete[] t;
00107             t = newT;
00108             currentCapacity = newSize * 2;
00109         }
00110     }
00111
00116     void push_back(T value) {
00117         if (currentSize == currentCapacity) {
00118             resize(currentSize * 2);
00119         }
00120         t[currentSize++] = value;
00121     }
00122
00128     int find(const T& value) const {
00129         for (size_t i = 0; i < currentSize; ++i) {
00130             if (t[i] == value) {
00131                 return i;
00132             }
00133         }
00134         return -1;
00135     }
00136
00141     class iterator {
00142     public:
00143         T *p;
00144         size_t idx;
00145
00152         iterator(T *p = nullptr, size_t idx = 0) : p(p), idx(idx) {}
00153
00162         T &operator*() { return p[idx]; }
00163
00167         iterator &operator++() {
00168             ++idx;
00169             return *this;
00170         }
00171
00176         bool operator!=(const iterator &other) const { return idx != other.idx; }
00177     };
00178
00183     class const_iterator {
00184     public:
00185         const T *p;
00186         size_t idx;
00187
00194         const_iterator(const T *p = nullptr, size_t idx = 0) : p(p), idx(idx) {}
00195
00204         const T &operator*() const { return p[idx]; }
00205
00209         const_iterator &operator++() {
00210             ++idx;
00211             return *this;
00212         }
00213
00218         bool operator!=(const const_iterator &other) const { return idx != other.idx; }

```

```
00219     };  
00220  
00224     virtual ~Tomb() {  
00225         delete[] t;  
00226     }  
00227 };  
00228  
00229 #endif
```

Tárgymutató

- ~Haromszog
 - Haromszog, [12](#)
- ~Kor
 - Kor, [18](#)
- ~Negyzet
 - Negyzet, [21](#)
- ~Sikidom
 - Sikidom, [32](#)
- ~Tomb
 - Tomb< T >, [38](#)
- at
 - Tomb< T >, [38](#), [39](#)
- begin
 - Tomb< T >, [39](#)
- capacity
 - Tomb< T >, [39](#)
- const_iterator
 - Tomb< T >::const_iterator, [8](#)
- createSikidom
 - Sikidom, [32](#)
- currentCapacity
 - Tomb< T >, [42](#)
- currentSize
 - Tomb< T >, [42](#)
- dst
 - Pont, [25](#)
 - pont.cpp, [43](#)
 - pont.h, [45](#)
- end
 - Tomb< T >, [39](#), [40](#)
- find
 - Tomb< T >, [40](#)
- Forgat
 - Pont, [25](#)
 - Sikidom, [32](#)
- getkp
 - Sikidom, [33](#)
- getp
 - Sikidom, [33](#)
- getx
 - Pont, [26](#)
- gety
 - Pont, [26](#)
- Haromszog, [10](#)
 - ~Haromszog, [12](#)
 - Haromszog, [12](#)
 - Kivul, [12](#)
 - operator<<, [14](#)
 - Rajtavan, [12](#)
 - Read, [13](#)
 - Terulet, [13](#)
 - Write, [13](#)
- idx
 - Tomb< T >::const_iterator, [10](#)
 - Tomb< T >::iterator, [16](#)
- IsOnTriangle
 - sikidom.cpp, [46](#)
 - sikidom.h, [48](#)
- iterator
 - Tomb< T >::iterator, [14](#)
- Kivul
 - Haromszog, [12](#)
 - Kor, [18](#)
 - Negyzet, [22](#)
 - Sikidom, [33](#)
- Kor, [16](#)
 - ~Kor, [18](#)
 - Kivul, [18](#)
 - Kor, [18](#)
 - operator<<, [20](#)
 - Rajtavan, [18](#)
 - Read, [19](#)
 - Terulet, [19](#)
 - Write, [19](#)
- kp
 - Sikidom, [36](#)
- Mozgat
 - Pont, [26](#)
 - Sikidom, [33](#)
- Negyzet, [20](#)
 - ~Negyzet, [21](#)
 - Kivul, [22](#)
 - Negyzet, [21](#)
 - operator<<, [23](#)
 - Rajtavan, [22](#)
 - Read, [22](#)
 - Terulet, [22](#)
 - Write, [23](#)

operator!=
 Pont, 26
 Tomb< T >::const_iterator, 9
 Tomb< T >::iterator, 15
 operator<<
 Haromszog, 14
 Kor, 20
 Negyzet, 23
 Pont, 29
 pont.cpp, 44
 Sikidom, 35
 sikidom.cpp, 47
 operator>>
 Pont, 29
 pont.cpp, 44
 Sikidom, 36
 sikidom.cpp, 47
 operator+
 Pont, 27
 operator++
 Tomb< T >::const_iterator, 9
 Tomb< T >::iterator, 15
 operator-
 Pont, 27
 operator=
 Pont, 27
 Sikidom, 33
 operator==
 Pont, 28
 operator[]
 Tomb< T >, 40, 41
 operator*
 Tomb< T >::const_iterator, 9
 Tomb< T >::iterator, 15

 p
 Sikidom, 36
 Tomb< T >::const_iterator, 10
 Tomb< T >::iterator, 16
 Pont, 23
 dst, 25
 Forgat, 25
 getx, 26
 gety, 26
 Mozgat, 26
 operator!=, 26
 operator<<, 29
 operator>>, 29
 operator+, 27
 operator-, 27
 operator=, 27
 operator==, 28
 Pont, 25
 setx, 28
 sety, 28
 x, 30
 y, 30
 pont.cpp
 dst, 43
 operator<<, 44
 operator>>, 44
 pont.h
 dst, 45
 push_back
 Tomb< T >, 41

 Rajtavan
 Haromszog, 12
 Kor, 18
 Negyzet, 22
 Sikidom, 34
 Read
 Haromszog, 13
 Kor, 19
 Negyzet, 22
 Sikidom, 34
 resize
 Tomb< T >, 41

 setx
 Pont, 28
 sety
 Pont, 28
 Sikidom, 30
 ~Sikidom, 32
 createSikidom, 32
 Forgat, 32
 getkp, 33
 getp, 33
 Kivul, 33
 kp, 36
 Mozgat, 33
 operator<<, 35
 operator>>, 36
 operator=, 33
 p, 36
 Rajtavan, 34
 Read, 34
 Sikidom, 31
 Tartalmazza, 34
 Terulet, 34
 Write, 34
 sikidom.cpp
 IsOnTriangle, 46
 operator<<, 47
 operator>>, 47
 sikidom.h
 IsOnTriangle, 48
 size
 Tomb< T >, 42
 src/pont.cpp, 43
 src/pont.h, 44, 45
 src/sikidom.cpp, 46
 src/sikidom.h, 47, 48
 src/tomb.hpp, 50

 t
 Tomb< T >, 42

Tartalmazza

Sikidom, [34](#)

Terület

Haromszog, [13](#)Kor, [19](#)Negyzet, [22](#)Sikidom, [34](#)

Tomb

Tomb< T >, [38](#)Tomb< T >, [37](#)~Tomb, [38](#)at, [38](#), [39](#)begin, [39](#)capacity, [39](#)currentCapacity, [42](#)currentSize, [42](#)end, [39](#), [40](#)find, [40](#)operator[], [40](#), [41](#)push_back, [41](#)resize, [41](#)size, [42](#)t, [42](#)Tomb, [38](#)Tomb< T >::const_iterator, [7](#)const_iterator, [8](#)idx, [10](#)operator!=, [9](#)operator++, [9](#)operator*, [9](#)p, [10](#)Tomb< T >::iterator, [14](#)idx, [16](#)iterator, [14](#)operator!=, [15](#)operator++, [15](#)operator*, [15](#)p, [16](#)

Write

Haromszog, [13](#)Kor, [19](#)Negyzet, [23](#)Sikidom, [34](#)

x

Pont, [30](#)

y

Pont, [30](#)