

Biblioteca de structuri de date

Generat de Doxygen 1.8.13

Cuprins

1	Indexul Claselor	1
1.1	Lista Claselor	1
2	Documentația Claselor	3
2.1	Referință la clasa (Template) <code>dsl::hashmap< key, value, hash, equal ></code>	3
2.1.1	Descriere Detaliată	3
2.1.2	Documentația Funcțiilor Membre	4
2.1.2.1	<code>begin()</code>	4
2.1.2.2	<code>clear()</code>	4
2.1.2.3	<code>empty()</code>	4
2.1.2.4	<code>end()</code>	4
2.1.2.5	<code>erase()</code>	5
2.1.2.6	<code>find()</code>	5
2.1.2.7	<code>insert()</code>	5
2.1.2.8	<code>size()</code>	5
2.2	Referință la clasa (Template) <code>dsl::heap< type, compare ></code>	5
2.2.1	Descriere Detaliată	6
2.2.2	Documentația pentru Constructori și Destructorii	6
2.2.2.1	<code>heap()</code>	6
2.2.3	Documentația Funcțiilor Membre	6
2.2.3.1	<code>clear()</code>	7
2.2.3.2	<code>empty()</code>	7
2.2.3.3	<code>pop()</code>	7
2.2.3.4	<code>push()</code>	7

2.2.3.5	size()	7
2.2.3.6	top()	7
2.3	Referință la structura <code>dsl::set< key, compare >::iterator</code>	8
2.3.1	Descriere Detaliată	8
2.3.2	Documentația Funcțiilor Membre	8
2.3.2.1	operator*()	8
2.3.2.2	operator++() [1/2]	9
2.3.2.3	operator++() [2/2]	9
2.3.2.4	operator--() [1/2]	9
2.3.2.5	operator--() [2/2]	9
2.3.2.6	operator->()	9
2.3.3	Documentația funcțiilor prietene sau înrudite	9
2.3.3.1	operator"!=	9
2.3.3.2	operator==	10
2.4	Referință la structura <code>dsl::hashmap< key, value, hash, equal >::iterator</code>	10
2.4.1	Descriere Detaliată	10
2.4.2	Documentația Funcțiilor Membre	11
2.4.2.1	operator*()	11
2.4.2.2	operator++() [1/2]	11
2.4.2.3	operator++() [2/2]	11
2.4.2.4	operator->()	11
2.4.3	Documentația funcțiilor prietene sau înrudite	11
2.4.3.1	operator"!=	11
2.4.3.2	operator==	12
2.5	Referință la structura <code>dsl::list< type >::iterator</code>	12
2.5.1	Descriere Detaliată	12
2.5.2	Documentația Funcțiilor Membre	13
2.5.2.1	operator*()	13
2.5.2.2	operator++() [1/2]	13
2.5.2.3	operator++() [2/2]	13

2.5.2.4	operator--() [1/2]	13
2.5.2.5	operator--() [2/2]	13
2.5.2.6	operator->()	13
2.5.3	Documentația funcțiilor prietene sau înrudite	14
2.5.3.1	operator"!=	14
2.5.3.2	operator==	14
2.6	Referință la clasa (Template) dsl::list< type >	14
2.6.1	Descriere Detaliată	15
2.6.2	Documentația pentru Constructori și Destructori	16
2.6.2.1	list() [1/2]	16
2.6.2.2	list() [2/2]	16
2.6.2.3	~list()	16
2.6.3	Documentația Funcțiilor Membre	16
2.6.3.1	back()	16
2.6.3.2	begin()	17
2.6.3.3	clear()	17
2.6.3.4	empty()	17
2.6.3.5	end()	17
2.6.3.6	erase()	17
2.6.3.7	front()	17
2.6.3.8	insert()	18
2.6.3.9	operator=()	18
2.6.3.10	size()	18
2.6.3.11	swap()	18
2.7	Referință la clasa (Template) dsl::set< key, compare >	18
2.7.1	Descriere Detaliată	19
2.7.2	Documentația pentru Constructori și Destructori	19
2.7.2.1	set()	19
2.7.3	Documentația Funcțiilor Membre	20
2.7.3.1	begin()	20
2.7.3.2	clear()	20
2.7.3.3	empty()	20
2.7.3.4	end()	20
2.7.3.5	erase()	20
2.7.3.6	find()	20
2.7.3.7	insert()	21
2.7.3.8	lower_bound()	21
2.7.3.9	operator=()	21
2.7.3.10	size()	21
2.7.3.11	swap()	21
2.7.3.12	upper_bound()	21

Capitolul 1

Indexul Claselor

1.1 Lista Claselor

Lista claselor, structurilor, uniunilor și interfețelor, cu scurte descrieri:

dsl::hashmap< key, value, hash, equal >	3
dsl::heap< type, compare >	5
dsl::set< key, compare >::iterator	8
dsl::hashmap< key, value, hash, equal >::iterator	10
dsl::list< type >::iterator	12
dsl::list< type >	14
dsl::set< key, compare >	18

Capitolul 2

Documentația Claselor

2.1 Referință la clasa (Template) `dsl::hashmap< key, value, hash, equal >`

```
#include <hashmap.h>
```

Membri

- struct `iterator`

Metode Publice

- `hashmap` (`size_t bucket_count`)
- `iterator begin` ()
- `iterator end` ()
- `iterator find` (`key id`)
- void `insert` (`const std::pair< key, value > &element`)
- void `erase` (`iterator it`)
- `size_t size` () const
- bool `empty` () const
- void `clear` ()

2.1.1 Descriere Detaliată

```
template<class key, class value, class hash = std::hash<key>, class equal = std::equal_to<key>>  
class dsl::hashmap< key, value, hash, equal >
```

Aceasta clasa este o implementare a unei tabele de dispersie ce folosește o cautare liniară pentru a rezolva coliziuni.

Un element reprezintă o pereche de tip cheie-valoare.

Elementele sunt organizate în "bucket"-uri pentru a se stoca valorile în funcție de "valoarea hash" a cheii.

Se folosește clasa `std::vector` pentru obiectul de tip "bucket".

Un articol ce explică sumar conceptul de tabelă hash se poate găsi la <https://infoarena.ro/tabele-hash-scurta-prezentare>.

Template Parameters

<i>key</i>	Tipul cheii unui element in tabela de dispersie.
<i>value</i>	Tipul valorii unui element in tabela de dispersie.
<i>hash</i>	O clasa ce reprezinta o functie unara ce accepta un parametru de tip "key", folosita pentru a genera "valoarea hash" a unei chei.
<i>equal</i>	O clasa ce reprezinta o functie binara ce accepta doi parametrii de tip "key", folosita pentru a verifica daca doua chei sunt egale.

2.1.2 Documentația Funcțiilor Membre

2.1.2.1 begin()

```
template<class key , class value , class hash = std::hash<key>, class equal = std::equal_↵
to<key>>
iterator dsl::hashmap< key, value, hash, equal >::begin ( ) [inline]
```

Returneaza un iterator ce reprezinta inceputul tabelii de dispersie, adica o referinta la primul element din tabela.

2.1.2.2 clear()

```
template<class key , class value , class hash = std::hash<key>, class equal = std::equal_↵
to<key>>
void dsl::hashmap< key, value, hash, equal >::clear ( ) [inline]
```

Sterge toate elementele din tabela de dispersie.

2.1.2.3 empty()

```
template<class key , class value , class hash = std::hash<key>, class equal = std::equal_↵
to<key>>
bool dsl::hashmap< key, value, hash, equal >::empty ( ) const [inline]
```

Verifica daca tabela de dispersie este goala.

2.1.2.4 end()

```
template<class key , class value , class hash = std::hash<key>, class equal = std::equal_↵
to<key>>
iterator dsl::hashmap< key, value, hash, equal >::end ( ) [inline]
```

Returneaza un iterator ce reprezinta sfarsitul tabelii de dispersie. Acest iterator nu trebuie accesat deoarece nu are o referinta la vreun element din tabela.

2.1.2.5 `erase()`

```
template<class key , class value , class hash = std::hash<key>, class equal = std::equal_↵
to<key>>
void dsl::hashmap< key, value, hash, equal >::erase (
    iterator it ) [inline]
```

Sterge elementul dat de iteratorul luat ca parametru. Daca iteratorul nu este valid, comportamentul acestei metode nu este definit.

2.1.2.6 `find()`

```
template<class key , class value , class hash = std::hash<key>, class equal = std::equal_↵
to<key>>
iterator dsl::hashmap< key, value, hash, equal >::find (
    key id ) [inline]
```

Returneaza un iterator la elementul ce are cheia data. Daca niciun element nu are cheia data, returneaza iteratorul ce marcheaza sfarsitul tablei.

2.1.2.7 `insert()`

```
template<class key , class value , class hash = std::hash<key>, class equal = std::equal_↵
to<key>>
void dsl::hashmap< key, value, hash, equal >::insert (
    const std::pair< key, value > & element ) [inline]
```

Insereaza un nou element in tabela de dispersie. Daca exista deja un element cu cheia data, valoarea sa nu este modificata.

2.1.2.8 `size()`

```
template<class key , class value , class hash = std::hash<key>, class equal = std::equal_↵
to<key>>
size_t dsl::hashmap< key, value, hash, equal >::size ( ) const [inline]
```

Returneaza numarul de elemente din tabela de dispersie.

Documentația pentru această clasă a fost generată din fișierul:

- `include/dsl/hashmap.h`

2.2 Referință la clasa (Template) `dsl::heap< type, compare >`

```
#include <heap.h>
```

Metode Publice

- `template<class Iter >`
 `heap` (Iter first, Iter last)
- `size_t size` () const
- `bool empty` () const
- `void push` (type value)
- `void pop` ()
- `type top` () const
- `void clear` ()

2.2.1 Descriere Detaliată

```
template<class type, class compare = std::less<type>>
class dsl::heap< type, compare >
```

Aceasta clasa este o implementare a unei cozi cu prioritati ce foloseste o structura de date de tip "heap".

Se foloseste clasa `std::vector` pentru a stoca elementele.

Un articol ce explica modul in care functioneaza structura de date de tip heap se poate gasi la <https://infoarena.ro/heapuri>.

Template Parameters

<i>type</i>	Tipul valorii unui element in structura de tip "heap".
<i>compare</i>	O clasa ce reprezinta o functie binara ce defineste o ordonare de tip "mai mic strict", folosita pentru a ordona elementele. Expresia <code>compare(a,b)</code> trebuie sa returneze true daca a ar trebui sa se afle inaintea lui b.

2.2.2 Documentația pentru Constructori și Destructori

2.2.2.1 heap()

```
template<class type , class compare = std::less<type>>
template<class Iter >
dsl::heap< type, compare >::heap (
    Iter first,
    Iter last ) [inline]
```

Construieste coada cu prioritati prin inserarea tuturor elementelor in intervalul [first,last) si sortarea structurii "heap".

2.2.3 Documentația Funcțiilor Membre

2.2.3.1 `clear()`

```
template<class type , class compare = std::less<type>>
void dsl::heap< type, compare >::clear ( ) [inline]
```

Goleste coada cu prioritati.

2.2.3.2 `empty()`

```
template<class type , class compare = std::less<type>>
bool dsl::heap< type, compare >::empty ( ) const [inline]
```

Verifica daca nu exista niciun element in coada cu prioritati.

2.2.3.3 `pop()`

```
template<class type , class compare = std::less<type>>
void dsl::heap< type, compare >::pop ( ) [inline]
```

Sterge primul element din coada, adica elementul cu prioritate maxima.

2.2.3.4 `push()`

```
template<class type , class compare = std::less<type>>
void dsl::heap< type, compare >::push (
    type value ) [inline]
```

Insereaza o noua valoare in coada cu prioritati.

2.2.3.5 `size()`

```
template<class type , class compare = std::less<type>>
size_t dsl::heap< type, compare >::size ( ) const [inline]
```

Returneaza numarul de elemente din coada cu prioritati.

2.2.3.6 `top()`

```
template<class type , class compare = std::less<type>>
type dsl::heap< type, compare >::top ( ) const [inline]
```

Returneaza valoarea primului element din coada, adica elementul cu prioritate maxima.

Documentația pentru această clasă a fost generată din fișierul:

- `include/dsl/heap.h`

2.3 Referință la structura `dsl::set< key, compare >::iterator`

```
#include <set.h>
```

Tipuri Publice

- using **iterator_category** = `std::bidirectional_iterator_tag`
- using **difference_type** = `std::ptrdiff_t`
- using **value_type** = `const key`
- using **pointer** = `const key *`
- using **reference** = `const key &`

Metode Publice

- **iterator** (`node *here, tree *structure`)
- reference **operator*** () const
- pointer **operator->** ()
- **iterator & operator++** ()
- **iterator operator++** (int)
- **iterator & operator--** ()
- **iterator operator--** (int)

Prieteni

- class **set**
- bool **operator==** (const **iterator** &a, const **iterator** &b)
- bool **operator!=** (const **iterator** &a, const **iterator** &b)

2.3.1 Descriere Detaliată

```
template<class key, class compare = std::less<key>>>
struct dsl::set< key, compare >::iterator
```

Aceasta clasa este iteratorul setului. Parcurgerea setului returneaza elementele in ordinea definita de clasa comparator a setului.

2.3.2 Documentația Funcțiilor Membre

2.3.2.1 **operator***()

```
template<class key , class compare = std::less<key>>>
reference dsl::set< key, compare >::iterator::operator* ( ) const [inline]
```

Obține o referință la un element din set.

2.3.2.2 operator++() [1/2]

```
template<class key , class compare = std::less<key>>
iterator& dsl::set< key, compare >::iterator::operator++ ( ) [inline]
```

Muta iteratorul la succesorul elementului curent din set.

2.3.2.3 operator++() [2/2]

```
template<class key , class compare = std::less<key>>
iterator dsl::set< key, compare >::iterator::operator++ (
    int ) [inline]
```

Muta iteratorul la succesorul elementului curent din set.

2.3.2.4 operator--() [1/2]

```
template<class key , class compare = std::less<key>>
iterator& dsl::set< key, compare >::iterator::operator-- ( ) [inline]
```

Muta iteratorul la predecesorul elementului curent din set.

2.3.2.5 operator--() [2/2]

```
template<class key , class compare = std::less<key>>
iterator dsl::set< key, compare >::iterator::operator-- (
    int ) [inline]
```

Muta iteratorul la predecesorul elementului curent din set.

2.3.2.6 operator->()

```
template<class key , class compare = std::less<key>>
pointer dsl::set< key, compare >::iterator::operator-> ( ) [inline]
```

Obține o referință la un element din set.

2.3.3 Documentația funcțiilor prietene sau înrudite**2.3.3.1 operator!=**

```
template<class key , class compare = std::less<key>>
bool operator!= (
    const iterator & a,
    const iterator & b ) [friend]
```

Verifica dacă doi iteratori nu sunt egali.

2.3.3.2 operator==

```
template<class key , class compare = std::less<key>>
bool operator== (
    const iterator & a,
    const iterator & b ) [friend]
```

Verifica daca doi iteratori sunt egali.

Documentația pentru această structură a fost generată din fișierul:

- include/dsl/set.h

2.4 Referință la structura `dsl::hashmap< key, value, hash, equal >::iterator`

```
#include <hashmap.h>
```

Tipuri Publice

- using **iterator_category** = std::forward_iterator_tag
- using **difference_type** = std::ptrdiff_t
- using **value_type** = std::pair< key, value >
- using **pointer** = std::pair< key, value > *
- using **reference** = std::pair< key, value > &

Metode Publice

- **iterator** (node here)
- reference **operator*** () const
- pointer **operator->** ()
- **iterator & operator++** ()
- **iterator operator++** (int)

Prieteni

- class **hashmap**
- bool **operator==** (const iterator &a, const iterator &b)
- bool **operator!=** (const iterator &a, const iterator &b)

2.4.1 Descriere Detaliată

```
template<class key, class value, class hash = std::hash<key>, class equal = std::equal_to<key>>
struct dsl::hashmap< key, value, hash, equal >::iterator
```

Această clasă este iteratorul tabelii de dispersie. Parcurgerea tabelii de dispersie returnează elementele într-o ordine aparent aleatorie.

2.4.2 Documentația Funcțiilor Membre

2.4.2.1 `operator*()`

```
template<class key , class value , class hash = std::hash<key>, class equal = std::equal_↵
to<key>>
reference dsl::hashmap< key, value, hash, equal >::iterator::operator* ( ) const [inline]
```

Obține o referință la un element din tabela de dispersie. Valoarea cheii este expusă doar ca nu trebuie modificată.

2.4.2.2 `operator++()` [1/2]

```
template<class key , class value , class hash = std::hash<key>, class equal = std::equal_↵
to<key>>
iterator& dsl::hashmap< key, value, hash, equal >::iterator::operator++ ( ) [inline]
```

Gaseste urmatoarea valoare in tabela de dispersie, sarind peste "bucket"-urile goale.

2.4.2.3 `operator++()` [2/2]

```
template<class key , class value , class hash = std::hash<key>, class equal = std::equal_↵
to<key>>
iterator dsl::hashmap< key, value, hash, equal >::iterator::operator++ (
    int ) [inline]
```

Gaseste urmatoarea valoare in tabela de dispersie, sarind peste "bucket"-urile goale.

2.4.2.4 `operator->()`

```
template<class key , class value , class hash = std::hash<key>, class equal = std::equal_↵
to<key>>
pointer dsl::hashmap< key, value, hash, equal >::iterator::operator-> ( ) [inline]
```

Obține o referință la un element din tabela de dispersie. Valoarea cheii este expusă doar ca nu trebuie modificată.

2.4.3 Documentația funcțiilor prietene sau înrudite

2.4.3.1 `operator"!="`

```
template<class key , class value , class hash = std::hash<key>, class equal = std::equal_↵
to<key>>
bool operator!= (
    const iterator & a,
    const iterator & b ) [friend]
```

Verifica dacă doi iteratori nu sunt egali.

2.4.3.2 operator==

```
template<class key , class value , class hash = std::hash<key>, class equal = std::equal_↵
to<key>>
bool operator== (
    const iterator & a,
    const iterator & b ) [friend]
```

Verifica daca doi iteratori sunt egali.

Documentația pentru această structură a fost generată din fișierul:

- include/dsl/hashmap.h

2.5 Referință la structura dsl::list< type >::iterator

```
#include <list.h>
```

Tipuri Publice

- using **iterator_category** = std::bidirectional_iterator_tag
- using **difference_type** = std::ptrdiff_t
- using **value_type** = type
- using **pointer** = type *
- using **reference** = type &

Metode Publice

- **iterator** (node *position)
- reference **operator*** () const
- pointer **operator->** ()
- **iterator & operator++** ()
- **iterator operator++** (int)
- **iterator & operator--** ()
- **iterator operator--** (int)

Prieteni

- class **list**
- bool **operator==** (const iterator &a, const iterator &b)
- bool **operator!=** (const iterator &a, const iterator &b)

2.5.1 Descriere Detaliată

```
template<class type>
struct dsl::list< type >::iterator
```

Aceasta clasa este iteratorul listei dublu inlantuite. Parcurgerea listei returneaza elementele in ordinea in care au fost inserate in lista.

2.5.2 Documentația Funcțiilor Membre

2.5.2.1 `operator*()`

```
template<class type >
reference dsl::list< type >::iterator::operator* ( ) const [inline]
```

Obține o referință la un element din lista.

2.5.2.2 `operator++()` [1/2]

```
template<class type >
iterator& dsl::list< type >::iterator::operator++ ( ) [inline]
```

Muta iteratorul la următorul element din lista.

2.5.2.3 `operator++()` [2/2]

```
template<class type >
iterator dsl::list< type >::iterator::operator++ (
    int ) [inline]
```

Muta iteratorul la următorul element din lista.

2.5.2.4 `operator--()` [1/2]

```
template<class type >
iterator& dsl::list< type >::iterator::operator-- ( ) [inline]
```

Muta iteratorul la elementul precedent din lista.

2.5.2.5 `operator--()` [2/2]

```
template<class type >
iterator dsl::list< type >::iterator::operator-- (
    int ) [inline]
```

Muta iteratorul la elementul precedent din lista.

2.5.2.6 `operator->()`

```
template<class type >
pointer dsl::list< type >::iterator::operator-> ( ) [inline]
```

Obține o referință la un element din lista.

2.5.3 Documentația funcțiilor prietene sau înrudite

2.5.3.1 operator!=

```
template<class type >
bool operator!= (
    const iterator & a,
    const iterator & b ) [friend]
```

Verifica daca doi iteratori nu sunt egali.

2.5.3.2 operator==

```
template<class type >
bool operator== (
    const iterator & a,
    const iterator & b ) [friend]
```

Verifica daca doi iteratori sunt egali.

Documentația pentru această structură a fost generată din fișierul:

- include/dsl/list.h

2.6 Referință la clasa (Template) `dsl::list< type >`

```
#include <list.h>
```

Membri

- struct `iterator`

Metode Publice

- `list` (const `list` &other)
- `list` & `operator=` (`list` other)
- `list` (`list` &&other) noexcept
- void `swap` (`list` &other)
- `~list` ()
- `iterator begin` ()
- `iterator end` ()
- `iterator insert` (`iterator` position, const type &value)
- `iterator erase` (`iterator` position)
- `size_t size` () const
- bool `empty` () const
- void `clear` ()
- type & `front` ()
- type & `back` ()

2.6.1 Descriere Detaliată

```
template<class type>  
class dsl::list< type >
```

Aceasta clasa este o implementare a unei liste dublu inlantuite.

Template Parameters

<i>type</i>	Tipul valorii unui element din lista.
-------------	---------------------------------------

2.6.2 Documentația pentru Constructori și Destructori

2.6.2.1 list() [1/2]

```
template<class type >
dsl::list< type >::list (
    const list< type > & other ) [inline]
```

Constructor de copiere, creeaza o copie a celeilalte liste.

2.6.2.2 list() [2/2]

```
template<class type >
dsl::list< type >::list (
    list< type > && other ) [inline], [noexcept]
```

Interschimba continutul acestei liste cu cel al listei de tip "rvalue".

2.6.2.3 ~list()

```
template<class type >
dsl::list< type >::~~list ( ) [inline]
```

Distruge obiectul lista.

2.6.3 Documentația Funcțiilor Membre

2.6.3.1 back()

```
template<class type >
type& dsl::list< type >::back ( ) [inline]
```

Returneaza o referinta la ultimul element al listei.

Comportamentul acestei functii apelate cand lista este goala este indefinit.

2.6.3.2 `begin()`

```
template<class type >
iterator dsl::list< type >::begin ( ) [inline]
```

Returnează un iterator ce reprezintă începutul listei.

2.6.3.3 `clear()`

```
template<class type >
void dsl::list< type >::clear ( ) [inline]
```

Golește lista.

2.6.3.4 `empty()`

```
template<class type >
bool dsl::list< type >::empty ( ) const [inline]
```

Verifica dacă nu există niciun element în lista.

2.6.3.5 `end()`

```
template<class type >
iterator dsl::list< type >::end ( ) [inline]
```

Returnează un iterator ce reprezintă sfârșitul listei. Acest iterator nu trebuie accesat deoarece nu are o referință la vreun element din lista.

2.6.3.6 `erase()`

```
template<class type >
iterator dsl::list< type >::erase (
    iterator position ) [inline]
```

Sterge elementul de la poziția specificată de iterator.

Returnează un iterator la elementul ce era succesorul elementului șters.

2.6.3.7 `front()`

```
template<class type >
type& dsl::list< type >::front ( ) [inline]
```

Returnează o referință la primul element al listei.

Comportamentul acestei funcții apelate când lista este goală este indefinit.

2.6.3.8 insert()

```
template<class type >
iterator dsl::list< type >::insert (
    iterator position,
    const type & value ) [inline]
```

Insereaza un nou element cu valoarea data inainte de elementul de la pozitia specificata de iterator.

Returneaza un iterator la noul element inserat.

2.6.3.9 operator=()

```
template<class type >
list& dsl::list< type >::operator= (
    list< type > other ) [inline]
```

Atribuire un nou continut listei, prin intermediul altei liste. Copiaza continutul celeilalte liste in lista curenta.

2.6.3.10 size()

```
template<class type >
size_t dsl::list< type >::size ( ) const [inline]
```

Returneaza numarul de elemente din lista.

2.6.3.11 swap()

```
template<class type >
void dsl::list< type >::swap (
    list< type > & other ) [inline]
```

Interschimba continutul acestei liste cu cel al listei date.

Documentația pentru această clasă a fost generată din fișierul:

- include/dsl/list.h

2.7 Referință la clasa (Template) dsl::set< key, compare >

```
#include <set.h>
```

Membri

- struct `iterator`

Metode Publice

- `set` (const `set` &other)
- `set` & `operator=` (`set` other)
- void `swap` (`set` &other)
- `iterator begin` ()
- `iterator end` ()
- void `insert` (const key &key_value)
- `iterator find` (const key &key_value)
- `iterator lower_bound` (const key &value)
- `iterator upper_bound` (const key &value)
- void `erase` (`iterator` to_erase)
- `size_t size` () const
- bool `empty` () const
- void `clear` ()

2.7.1 Descriere Detaliată

```
template<class key, class compare = std::less<key>>
class dsl::set< key, compare >
```

Aceasta clasa este o implementare a unui set ordonat ce foloseste o structura de date de tip "treap".

Elementele din set sunt ordonate dupa valoarea cheii lor.

Un articol ce explica modul in care functioneaza structura de date de tip "treap" se poate gasi la <https://www.infoarena.ro/treapuri>.

Template Parameters

<i>key</i>	Tipul valorii unui element din set.
<i>compare</i>	O clasa ce reprezinta o functie binara ce defineste o ordonare de tip "mai mic strict", folosita pentru a ordona elementele. Expresia <code>compare(a,b)</code> trebuie sa returneze true daca a ar trebui sa se afle inaintea lui b.

2.7.2 Documentația pentru Constructori și Destructori

2.7.2.1 `set()`

```
template<class key , class compare = std::less<key>>
dsl::set< key, compare >::set (
    const set< key, compare > & other ) [inline]
```

Constructor de copiere, creeaza o copie a celui alt set.

2.7.3 Documentația Funcțiilor Membre

2.7.3.1 begin()

```
template<class key , class compare = std::less<key>>
iterator dsl::set< key, compare >::begin ( ) [inline]
```

Returnează un iterator ce reprezintă începutul setului.

2.7.3.2 clear()

```
template<class key , class compare = std::less<key>>
void dsl::set< key, compare >::clear ( ) [inline]
```

Golește setul.

2.7.3.3 empty()

```
template<class key , class compare = std::less<key>>
bool dsl::set< key, compare >::empty ( ) const [inline]
```

Verifică dacă setul este gol.

2.7.3.4 end()

```
template<class key , class compare = std::less<key>>
iterator dsl::set< key, compare >::end ( ) [inline]
```

Returnează un iterator ce reprezintă sfârșitul setului. Acest iterator nu trebuie accesat deoarece nu are o referință la vreun element din set.

2.7.3.5 erase()

```
template<class key , class compare = std::less<key>>
void dsl::set< key, compare >::erase (
    iterator to_erase ) [inline]
```

Sterge elementul de la poziția specificată de iterator.

2.7.3.6 find()

```
template<class key , class compare = std::less<key>>
iterator dsl::set< key, compare >::find (
    const key & key_value ) [inline]
```

Returnează un iterator la elementul cu cheia dată. Dacă elementul nu este găsit, returnează iteratorul ce marchează sfârșitul setului.

2.7.3.7 `insert()`

```
template<class key , class compare = std::less<key>>
void dsl::set< key, compare >::insert (
    const key & key_value ) [inline]
```

Insereaza un nou element cu valoarea data in set.

2.7.3.8 `lower_bound()`

```
template<class key , class compare = std::less<key>>
iterator dsl::set< key, compare >::lower_bound (
    const key & value ) [inline]
```

Returneaza un iterator la primul element a carui valoare nu preceda valoarea data. Elementul poate avea o valoare egala, sau o valoare ce "merge dupa" valoarea data.

2.7.3.9 `operator=()`

```
template<class key , class compare = std::less<key>>
set& dsl::set< key, compare >::operator= (
    set< key, compare > other ) [inline]
```

Atribuire un nou continut setului, prin intermediul altui set. Copiaza continutul celui alt set in setul curent.

2.7.3.10 `size()`

```
template<class key , class compare = std::less<key>>
size_t dsl::set< key, compare >::size ( ) const [inline]
```

Returneaza numarul de elemente din set.

2.7.3.11 `swap()`

```
template<class key , class compare = std::less<key>>
void dsl::set< key, compare >::swap (
    set< key, compare > & other ) [inline]
```

Interschimba continutul acestui set cu cel al setului dat.

2.7.3.12 `upper_bound()`

```
template<class key , class compare = std::less<key>>
iterator dsl::set< key, compare >::upper_bound (
    const key & value ) [inline]
```

Returneaza un iterator la primul element a carui valoare "merge dupa" valoarea data.

Documentația pentru această clasă a fost generată din fișierul:

- `include/dsl/set.h`

Glosar

~list
 dsl::list, 16

back
 dsl::list, 16

begin
 dsl::hashmap, 4
 dsl::list, 16
 dsl::set, 20

clear
 dsl::hashmap, 4
 dsl::heap, 6
 dsl::list, 17
 dsl::set, 20

dsl::hashmap
 begin, 4
 clear, 4
 empty, 4
 end, 4
 erase, 4
 find, 5
 insert, 5
 size, 5

dsl::hashmap< key, value, hash, equal >, 3

dsl::hashmap< key, value, hash, equal >::iterator, 10

dsl::hashmap::iterator
 operator!=, 11
 operator*, 11
 operator++, 11
 operator->, 11
 operator==, 11

dsl::heap
 clear, 6
 empty, 7
 heap, 6
 pop, 7
 push, 7
 size, 7
 top, 7

dsl::heap< type, compare >, 5

dsl::list
 ~list, 16
 back, 16
 begin, 16
 clear, 17
 empty, 17
 end, 17
 erase, 17
 front, 17
 insert, 17
 list, 16
 operator=, 18
 size, 18
 swap, 18

dsl::list< type >, 14

dsl::list< type >::iterator, 12

dsl::list::iterator
 operator!=, 14
 operator*, 13
 operator++, 13
 operator->, 13
 operator--, 13
 operator==, 14

dsl::set
 begin, 20
 clear, 20
 empty, 20
 end, 20
 erase, 20
 find, 20
 insert, 20
 lower_bound, 21
 operator=, 21
 set, 19
 size, 21
 swap, 21
 upper_bound, 21

dsl::set< key, compare >, 18

dsl::set< key, compare >::iterator, 8

dsl::set::iterator
 operator!=, 9
 operator*, 8
 operator++, 8, 9
 operator->, 9
 operator--, 9
 operator==, 9

empty
 dsl::hashmap, 4
 dsl::heap, 7
 dsl::list, 17
 dsl::set, 20

end
 dsl::hashmap, 4
 dsl::list, 17
 dsl::set, 20

erase
 dsl::hashmap, 4

dsl::list, 17
 dsl::set, 20

find
 dsl::hashmap, 5
 dsl::set, 20

front
 dsl::list, 17

heap
 dsl::heap, 6

insert
 dsl::hashmap, 5
 dsl::list, 17
 dsl::set, 20

list
 dsl::list, 16

lower_bound
 dsl::set, 21

operator!=
 dsl::hashmap::iterator, 11
 dsl::list::iterator, 14
 dsl::set::iterator, 9

operator*
 dsl::hashmap::iterator, 11
 dsl::list::iterator, 13
 dsl::set::iterator, 8

operator++
 dsl::hashmap::iterator, 11
 dsl::list::iterator, 13
 dsl::set::iterator, 8, 9

operator->
 dsl::hashmap::iterator, 11
 dsl::list::iterator, 13
 dsl::set::iterator, 9

operator--
 dsl::list::iterator, 13
 dsl::set::iterator, 9

operator=
 dsl::list, 18
 dsl::set, 21

operator==
 dsl::hashmap::iterator, 11
 dsl::list::iterator, 14
 dsl::set::iterator, 9

pop
 dsl::heap, 7

push
 dsl::heap, 7

set
 dsl::set, 19

size
 dsl::hashmap, 5
 dsl::heap, 7
 dsl::list, 18

 dsl::set, 21

swap
 dsl::list, 18
 dsl::set, 21

top
 dsl::heap, 7

upper_bound
 dsl::set, 21