

Cekla

Készítette Doxygen 1.7.1

Tue Sep 20 2011 14:13:44

Tartalomjegyzék

1. Cékla C++ könyvtár deklaratív C++-hoz	1
2. Modulmutató	1
2.1. Modulok	1
3. Osztálymutató	2
3.1. Osztálylista	2
4. Fájlmutató	2
4.1. Fájllista	2
5. Modulok dokumentációja	2
5.1. CéklaList	2
5.1.1. Részletes leírás	3
5.1.2. Függvények dokumentációja	3
5.1.3. Változók dokumentációja	5
5.2. CéklaFunctional	5
5.2.1. Részletes leírás	5
6. Osztályok dokumentációja	6
6.1. list osztályreferencia	6
6.1.1. Részletes leírás	6
6.1.2. Konstruktorkok és destruktorkok dokumentációja	6
6.1.3. Tagfüggvények dokumentációja	7
7. Fájlok dokumentációja	7
7.1. cekla.h fájlreferencia	7
7.1.1. Részletes leírás	9

1. Cékla C++ könyvtár deklaratív C++-hoz

Installálás

Helyezzük a `cekla.h` fájlt a forrásfájlokkal egy könyvtárba, vagy a header fájlok keresési útvonalába. Vagy GCC esetén a `CPATH` környezeti változót állítsuk a `cekla.h`-t tartalmazó könyvtárra, például `export CPATH=/opt/cekla/include`. A konfiguráció teszteléséhez egy rövid példa:

```
#include "cekla.h"
int is_empty(const list L) { return L == nil; }
int main() {
    writeln(is_empty(nil));
}
```

```
writeln(is_empty(cons(10, nil)));  
}
```

Használat

- Listákhoz (lásd a [CeklaList](#) modult) vagy függvénytípusokhoz (lásd a [CeklaFunctional](#) modult), használjuk az `include "cekla.h"` direktívát.
- Nem-deklaratív függvényhívások is megengedettek pl. hibakereséshez például használhatjuk a `writeln`-t utasítások között
- Gépeljük be a `help;` parancsot a `Cekla` értelmezőjében a megengedett szintaxishoz.

Fontos makrók

- `NDEBUG`: ha definiáljuk, a `write`, `writeln` hívások nem írnak ki debug információt (a hívás sorszámát a forráskódban).
- `ENABLE_SHARED_PTR`: bekapcsolja a szemétgyűjtést megakadályozva a memóriaszivárgást, de lehetetlenné teszi a GCC-nek, hogy optimalizálja a jobbrekurzív függvényeket.

Szerző

Copyright (C) 2011, BME Deklaratív Programozás

<http://dp.iit.bme.hu/> \$Revision: 320 \$

2. Modulmutató

2.1. Modulok

A modulok listája:

CeklaList	2
CeklaFunctional	5

3. Osztálymutató

3.1. Osztálylista

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

list (Egészek listája)	6
--------------------------------	----------

4. Fájlmutató

4.1. Fájllista

Az összes dokumentált fájl listája rövid leírásokkal:

[cekla.h](#) (Cekla C++ könyvtár)

7

5. Modulok dokumentációja

5.1. CeklaList

Listakezelés C++-ban.

Osztályok

- class [list](#)
Egészek listája.

Függvények

- [list cons](#) (int Head, const [list](#) Tail)
Visszaad egy új listát, aminek első eleme Head, a farka a Tail lista.
- [int hd](#) (const [list](#) L)
Visszaadja a nemüres L lista fejét.
- [list tl](#) (const [list](#) L)
Visszaadja a nemüres L lista farkát.
- `template<typename any_type >`
`void write (const any_type &X)`
Kírja X-et a standard kimenetre.
- `template<typename any_type >`
`void writeln (const any_type &X)`
Kírja X-et a standard kimenetre, és egy újsorjelet.
- [list l](#) ()
Visszaadja a paramétereiből alkotott listát.
- [list l](#) (int E)
Visszaadja a paramétereiből alkotott listát.
- [list l](#) (int E1, int E2)
Visszaadja a paramétereiből alkotott listát.
- [list l](#) (int E1, int E2, int E3)
Visszaadja a paramétereiből alkotott listát.
- [list l](#) (int E1, int E2, int E3, int E4)
Visszaadja a paramétereiből alkotott listát.

- `list l` (int E1, int E2, int E3, int E4, int E5)
Visszaadja a paramétereiből alkotott listát.
- `list l` (int E1, int E2, int E3, int E4, int E5, int E6)
Visszaadja a paramétereiből alkotott listát.
- `list l` (int E1, int E2, int E3, int E4, int E5, int E6, int E7)
Visszaadja a paramétereiből alkotott listát.
- `list l` (int E1, int E2, int E3, int E4, int E5, int E6, int E7, int E8)
Visszaadja a paramétereiből alkotott listát.
- `list l` (int E1, int E2, int E3, int E4, int E5, int E6, int E7, int E8, int E9)
Visszaadja a paramétereiből alkotott listát.
- `list l` (int E1, int E2, int E3, int E4, int E5, int E6, int E7, int E8, int E9, int E10)
Visszaadja a paramétereiből alkotott listát.

Változók

- `const list nil = ""`
Az üres lista.

5.1.1. Részletes leírás

Listakezelés C++-ban.

5.1.2. Függvények dokumentációja

5.1.2.1. `list cons (int Head, const list Tail)`

Visszaad egy új listát, aminek első eleme Head, a farka a Tail lista.

Paraméterek

Head Az elem lesz az új lista első eleme.

Tail A lista lesz az új lista többi eleme.

Visszatérési érték

A felépített lista.

Futási idő: O(1).

Például: `cons('H', cons('e', cons('l', cons('l', cons('o', nil)))))) == "Hello"`.

5.1.2.2. int hd (const list L)

Visszaadja a nemüres L lista fejét.

Visszatérési érték

L első eleme.

Futási idő: O(1).

Például: `hd("Hello") == 'H'`.

5.1.2.3. list l(int E1, int E2, int E3, int E4, int E5, int E6, int E7, int E8, int E9, int E10)

Visszaadja a paramétereiből alkotott listát.

Például: `const list L = l(10, 20, 30, 40, 50, 60, 70, 80, 90, 0);`

5.1.2.4. list tl (const list L)

Visszaadja a nemüres L lista farkát.

Visszatérési érték

L lista elemei, az első elem kivételével.

Futási idő: O(1).

Például: `tl("Hello") == "ello"`.

5.1.2.5. template<typename any_type > void write (const any_type & X)

Kírja X-et a standard kimenetre.

Paraméterek

X Típusa lehet int, sztringkonstans vagy lista. Ha a lista nemcsak 32..126 közötti számokat tartalmaz, egész listaként íródik ki (pl. [10, 20, 30]), különben karakterkód-listaként (pl. "hello").

5.1.2.6. template<typename any_type > void writeln (const any_type & X)

Kírja X-et a standard kimenetre, és egy újsorjelet.

Lásd még

[write](#).

5.1.3. Változók dokumentációja

5.1.3.1. const list nil = ""

Az üres lista.

A `nil == ""` teljesül. Használható egy lista vizsgálatára, hogy üres-e: `tl("e") == nil`, vagy lista építésére: `cons('H', cons('e', cons('l', cons('l', cons('o', nil)))))` `== "Hello"`.

5.2. CeklaFunctional

Típusok magasabbrendű függvényekhez.

Típusdefiníciók

- `typedef int(* fun1)(int)`
Egyparaméteres függvénytípus magasabbrendű függvényekhez.
- `typedef int(* fun2)(int, int)`
Kétparaméteres függvénytípus magasabbrendű függvényekhez.

5.2.1. Részletes leírás

Típusok magasabbrendű függvényekhez. Egész-függvények átadására használhatóak. Például az alábbi program kimenete 1:

```
#include "cekla.h"
// Returns true if Predicate(X) is true for some element X of list L
int contains(const fun1 Predicate, const list L) {
    if (L == nil) return 0;
    else if (Predicate(hd(L))) return 1;
    else return contains(Predicate, tl(L));
}
int even(const int x) { return x % 2 == 0; }
int main() {
    const list L = cons(1, cons(2, cons(3, nil)));
    write(contains(even, L)); // prints if L contains even number
}
```

6. Osztályok dokumentációja

6.1. list osztályreferencia

Egészek listája.

```
#include <cekla.h>
```

Publikus tagfüggvények

- `list (const char *S) throw (std::logic_error)`
Felépíti a karakterkódok listáját.
- `list (std::initializer_list< int > S)`
Felépíti listát az inicializáló lista számaiból.
- `bool operator== (const list &Rhs) const`
Összehasonlít két listát.
- `bool operator!= (const list &Rhs) const`
Összehasonlít két listát.

6.1.1. Részletes leírás

Egészek listája. Sztring (C-nyelvű karaktertömb, pl. "hello") karakterkódok listájának tekintendő a lezáró '\0' nélkül. Például: `cons(72, cons(101, cons(108, cons(108, cons(111, nil)))) == "Hello"`.

6.1.2. Konstruktorkok és destruktorok dokumentációja

6.1.2.1. `list::list (const char * S) throw (std::logic_error)`

Felépíti a karakterkódok listáját.

Kivételek

`std::logic_error` ha a paraméter NULL pointer.

Hasznos sztring implicit konverziójához, például `const list L1 = "Hello", L2 = "";`, vagy `X = tl("Hello")` rövidíti az alábbi: `X = tl(cons('H', cons('e', cons('l', cons('l', cons('o', nil))))))`.

6.1.2.2. `list::list (std::initializer_list< int > S)`

Felépíti listát az inicializáló lista számaiból.

Csak C++0x és `ENABLE_INITIALIZER_LIST` definálása esetén elérhető. Például: `tl({10, 20, 30})`.

6.1.3. Tagfüggvények dokumentációja

6.1.3.1. `bool list::operator!= (const list & Rhs) const`

Összehasonlít két listát.

Visszatérési érték

Igaz (nem nulla), ha a két lista nem egyezik.

Lásd még

[operator==](#)

6.1.3.2. `bool list::operator==(const list & Rhs) const`

Összehasonlít két listát.

Visszatérési érték

Igaz (nem nulla), ha a két lista egyezik.

Ez a dokumentáció az osztályról a következő fájl alapján készült:

- [cekla.h](#)

7. Fájlok dokumentációja

7.1. `cekla.h` fájlreferencia

Cekla C++ könyvtár.

Osztályok

- class [list](#)
Egészek listája.

Típusdefiníciók

- typedef int(* [fun1](#))(int)
Egyparaméteres függvénytípus magasabbrendű függvényekhez.
- typedef int(* [fun2](#))(int, int)
Kétparaméteres függvénytípus magasabbrendű függvényekhez.

Függvények

- [list cons](#) (int Head, const [list](#) Tail)
Visszaad egy új listát, aminek első eleme Head, a farka a Tail lista.
- int [hd](#) (const [list](#) L)

Visszaadja a nemüres *L* lista fejét.

- `list tl` (const `list L`)

Visszaadja a nemüres *L* lista farkát.

- `template<typename any_type >`
`void write` (const `any_type &X`)

Kírja *X*-et a standard kimenetre.

- `template<typename any_type >`
`void writeln` (const `any_type &X`)

Kírja *X*-et a standard kimenetre, és egy újsorjelet.

- `list l` ()

Visszaadja a paramétereiből alkotott listát.

- `list l` (int *E*)

Visszaadja a paramétereiből alkotott listát.

- `list l` (int *E1*, int *E2*)

Visszaadja a paramétereiből alkotott listát.

- `list l` (int *E1*, int *E2*, int *E3*)

Visszaadja a paramétereiből alkotott listát.

- `list l` (int *E1*, int *E2*, int *E3*, int *E4*)

Visszaadja a paramétereiből alkotott listát.

- `list l` (int *E1*, int *E2*, int *E3*, int *E4*, int *E5*)

Visszaadja a paramétereiből alkotott listát.

- `list l` (int *E1*, int *E2*, int *E3*, int *E4*, int *E5*, int *E6*)

Visszaadja a paramétereiből alkotott listát.

- `list l` (int *E1*, int *E2*, int *E3*, int *E4*, int *E5*, int *E6*, int *E7*)

Visszaadja a paramétereiből alkotott listát.

- `list l` (int *E1*, int *E2*, int *E3*, int *E4*, int *E5*, int *E6*, int *E7*, int *E8*)

Visszaadja a paramétereiből alkotott listát.

- `list l` (int *E1*, int *E2*, int *E3*, int *E4*, int *E5*, int *E6*, int *E7*, int *E8*, int *E9*)

Visszaadja a paramétereiből alkotott listát.

- `list l` (int *E1*, int *E2*, int *E3*, int *E4*, int *E5*, int *E6*, int *E7*, int *E8*, int *E9*, int *E10*)

Visszaadja a paramétereiből alkotott listát.

Változók

- const `list nil` = ""

Az üres lista.

7.1.1. Részletes leírás

Cekla C++ könyvtár.

Tárgymutató

cekla.h, [7](#)
CeklaFunctional, [5](#)
CeklaList, [2](#)
 cons, [3](#)
 hd, [4](#)
 l, [4](#)
 nil, [5](#)
 tl, [4](#)
 write, [4](#)
 writeln, [5](#)
cons
 CeklaList, [3](#)

hd
 CeklaList, [4](#)

l
 CeklaList, [4](#)
list, [6](#)
 list, [6](#)
 operator==, [7](#)

nil
 CeklaList, [5](#)

operator==
 list, [7](#)

tl
 CeklaList, [4](#)

write
 CeklaList, [4](#)
writeln
 CeklaList, [5](#)