Dinamikus string - Dokumentáció

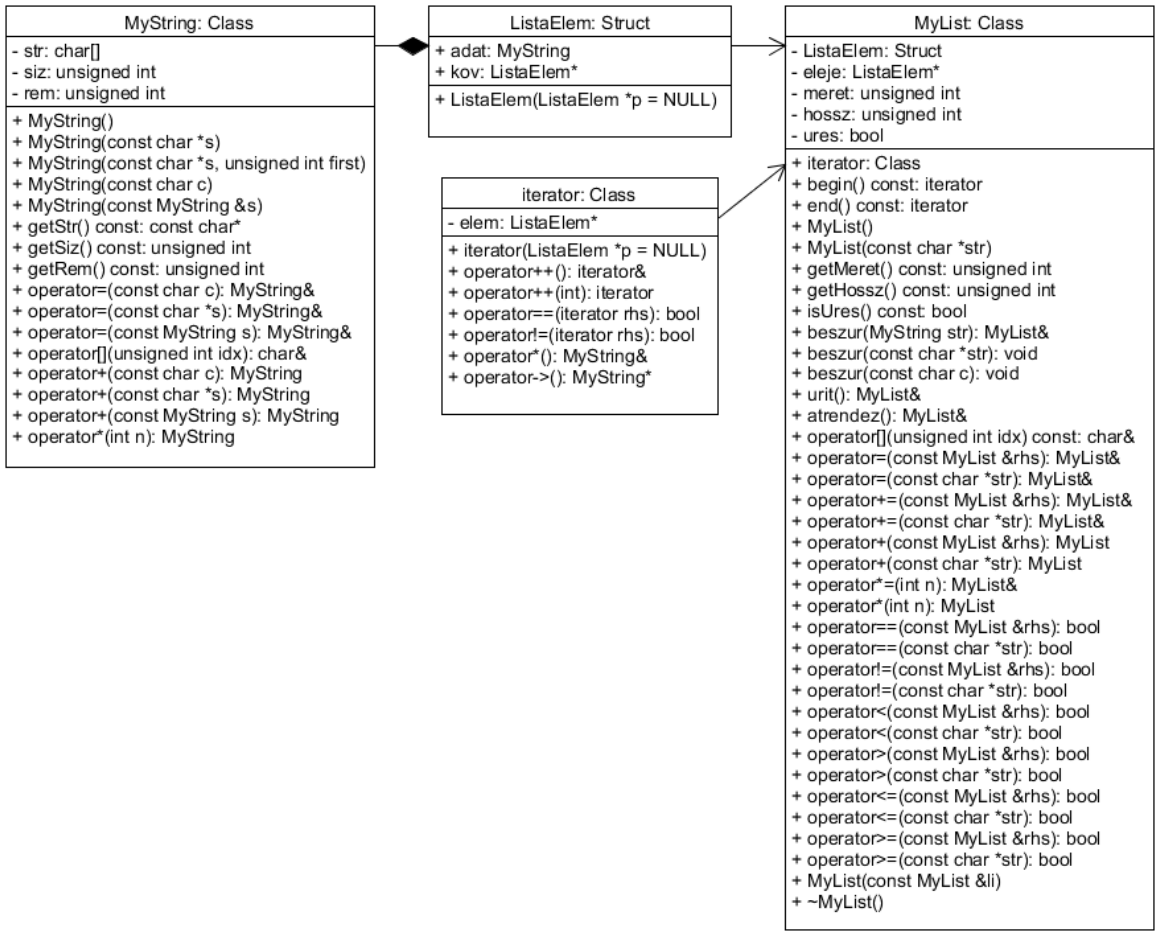
Pontosított feladatspecifikáció:

Ez egy olyan dinamikus string, ami a karaktereket egy 20 karakteres tárolókból épített láncolt lista tárolja. A feladat magában foglalja az értelmes műveletek operátorainak átdefiniálását. Az osztály rendelkezni fog iterátorral is. A program futása annyiból fog állni, hogy képesek leszünk akármilyen hosszúságú karaktersorozatot beolvasni az adatszerkezetbe. Ellenőrzésképp ki is írja azt, valamint a hosszát és a 20 karakteres tárolók számát.

Megvalósítandó operátorok:

- indexelés (adott karaktert adja vissza)
- értékadás
- összeadás (két string konkatenálása)
- szorzás (paraméterként adott string tartalmát megtöbbszörözi)
- összehasonlító operátorok (== , != , < , > , <= , >=)
- kiíró << operátor

Osztálydiagram:



Rövid leírás:

Van egy MyString nevű osztály, ami 20 karaktert tárol (set.h-ban állítható).
MyList nevű osztály valósítja meg a láncolt listát, amiben van a ListaElem osztály, ami tárolja a MyString objektumokat.
MyList-ben van az iterator osztály is, amivel a ListaElem-eken tudunk végigiterálni.
Minden értelmes operátor meg van valósítva, ezért használható úgy, akár egy std::string.
Program indításakor a paraméterként kapott karaktersorozatokból MyList-eket csinál. Kiírja azokat, hosszukat, és a tárolók számát.

Tesztelés:

A tesztesetek gtest_lite segítségével vannak megoldva, amelyek a main.cpp-ben találhatók. Minden teszteset lefut a program indításakor és nem keletkezik hiba.

Memóriaszivárgás:

A program nem rendelkezik memóriaszivárgással, ugyanis a MyList destruktora felszabadítja a dinamikusan foglalt ListaElem-eket. Ezt a memtrace is alátámasztja.

Class Documentation

MyList::iterator Class Reference

MyList iterátora.
#include <MyList.h>

Public Member Functions

- **iterator** (**ListaElem** *p=NULL)
- **iterator** & **operator++** ()
  MyList *következö elemére lép (pre-inkremens)*
- **iterator operator++** (int)
  MyList *következö elemére lép (post-inkremens)*
- bool **operator==** (**iterator** rhs)
  Összehasonlítja (egyenlö) a paraméterként adott iterátorral.
- bool **operator!=** (**iterator** rhs)
  Összehasonlítja (nem egyenlö) a paraméterként adott iterátorral.
- **MyString** & **operator*** ()
- **MyString** * **operator->** ()

Private Attributes

- **ListaElem** * **elem**
  Pointert tárol ListaElem-re.

Detailed Description

MyList iterátora.

Constructor & Destructor Documentation

MyList::iterator::iterator (ListaElem * *p* = NULL)[inline]

Iterátor konstruktora: Eltárolja a paraméterként adott pointert

Parameters:

p	- **ListaElem** pointer

Member Function Documentation

MyString& MyList::iterator::operator* ()[inline]

Iterátor indirekciója (*)

Returns:

MyString referenciát ad vissza

MyString* MyList::iterator::operator-> ()[inline]

Iterátor indirekciója (->)

Returns:

MyString pointert ad vissza

The documentation for this class was generated from the following file:

- MyList.h

MyList::ListaElem Struct Reference

MyList-ben tárolt **ListaElem**.

Public Member Functions

- **ListaElem** (**ListaElem** *p=NULL)
  ListaElem *konstruktora.*

Public Attributes

- **MyString adat**
  ListaElem-ben tárolt ***MyString****.*
- **ListaElem * kov**
  Pointer a következö ListaElem-re.

Detailed Description

MyList-ben tárolt **ListaElem**.

The documentation for this struct was generated from the following file:

- MyList.h

MyList Class Reference

#include <MyList.h>

Classes

- class **iterator**
- struct **ListaElem**

Public Member Functions

- **iterator begin** () const
  *Iterátort ad vissza a **MyList** elejére.*
- **iterator end** () const
  *Iterátort ad vissza a **MyList** végére.*
- **MyList** ()
- **MyList** (const char *str)
- unsigned int **getMeret** () const
  Visszaadja a MyList-ben tárolt MyString-ek számát
- unsigned int **getHossz** () const
  *Visszaadja a **MyList** hosszát*
- bool **isUres** () const
  *Visszaadja, hogy üres-e a **MyList**.*
- **MyList** & **beszur** (**MyString** str)
- void **beszur** (const char *str)
- void **beszur** (const char c)
- **MyList** & **urit** ()
- **MyList** & **atrendez** ()
- char & **operator[]** (unsigned int idx) const
- **MyList** & **operator=** (const **MyList** &rhs)
- **MyList** & **operator=** (const char *str)
- **MyList** & **operator+=** (const **MyList** &rhs)
- **MyList** & **operator+=** (const char *str)
- **MyList operator+** (const **MyList** &rhs)
- **MyList operator+** (const char *str)
- **MyList** & **operator*=** (int n)
- **MyList operator*** (int n)
- bool **operator==** (const **MyList** &rhs)
- bool **operator==** (const char *str)
- bool **operator!=** (const **MyList** &rhs)
- bool **operator!=** (const char *str)
- bool **operator<** (const **MyList** &rhs)
- bool **operator<** (const char *str)
- bool **operator>** (const **MyList** &rhs)
- bool **operator>** (const char *str)
- bool **operator<=** (const **MyList** &rhs)
- bool **operator<=** (const char *str)
- bool **operator>=** (const **MyList** &rhs)
- bool **operator>=** (const char *str)
- **MyList** (const **MyList** &li)
  Másolókonstruktor.

- **~MyList** ()
  Destruktor felszabadítja a listaelemeket.

Private Attributes

- **ListaElem** * **eleje**
  Pointer az elsö ListaElem-re.
- unsigned int **meret**
  MyList-ben tárolt MyString-ek száma.
- unsigned int **hossz**
  ***MyList** hossza.*
- bool **ures**
  *Tárolja, hogy üres-e a **MyList**.*

Detailed Description

MyString objektumokat tárol láncolt listában

Constructor & Destructor Documentation

MyList::MyList ()[inline]

Paraméter nélkül hívható konstruktor. Üres MyList-et hoz létre.

MyList::MyList (const char * *str*)

Konstruktor: MyList-et hoz létre karaktertömbböl

Parameters:

str	- karaktertömb

MyList::~MyList ()

Destruktor felszabadítja a listaelemeket

Member Function Documentation

MyList & MyList::atrendez ()

Amennyiben a MyList-ben vannak olyan MyString-ek, amelyek nincsenek feltöltve teljesen és az nem az **MyString**, akkor átrendezi a karaktereket, hogy ne legyenek üres helyek a MyString-ekben

MyList & MyList::beszur (MyString *str*)

MyList-be beszúr MyString-et

Parameters:

str	- **MyString**

void MyList::beszur (const char * *str*)[inline]

MyList-be beszúr karaktertömböt MyString-ként

Parameters:

str	- karaktertömb

void MyList::beszur (const char *c*)[inline]

MyList-be beszúr karaktert MyString-ként

Parameters:

c	- karakter

bool MyList::operator!= (const MyList & *rhs*)

Összehasonlítja (nem egyenlö) a paraméterként adott MyList-tel

Parameters:

rhs	- **MyList**

bool MyList::operator!= (const char * *str*)

Összehasonlítja (nem egyenlö) a paraméterként adott karaktertömbbel

Parameters:

str	- karaktertömb

MyList MyList::operator* (int *n*)

Meglévõ **MyList** tartalmát többszörözi meg

Parameters:

n	- szorzó

Returns:

új szorzat MyList-et ad vissza

MyList & MyList::operator*= (int *n*)

Meglévõ **MyList** tartalmát többszörözi meg

Parameters:

n	- szorzó

Returns:

meglévö MyList-et adja vissza referenciaként megszorozva

MyList MyList::operator+ (const MyList & *rhs*)

Meglévõ MyList-hez füz MyList-et

Parameters:

rhs	- **MyList**

Returns:

új összeg MyList-et ad vissza

MyList MyList::operator+ (const char * *str*)

Meglévõ MyList-hez füz karaktertömböt

Parameters:

str	- karaktertömb

Returns:

új összeg MyList-et ad vissza

MyList & MyList::operator+= (const MyList & *rhs*)

Meglévõ MyList-hez füz MyList-et

Parameters:

rhs	- **MyList**

Returns:

meglévö MyList-et adja vissza hozzáfüzve

MyList & MyList::operator+= (const char * *str*)

Meglévõ MyList-hez füz karaktertömböt

Parameters:

str	- karaktertömb

Returns:

meglévö MyList-et adja vissza hozzáfüzve

bool MyList::operator< (const MyList & *rhs*)

Összehasonlítja (kisebb) a paraméterként adott MyList-tel

Parameters:

rhs	- **MyList**

bool MyList::operator< (const char * *str*)

Összehasonlítja (kisebb) a paraméterként adott karaktertömbbel

Parameters:

str	- karaktertömb

bool MyList::operator<= (const MyList & *rhs*)

Összehasonlítja (kisebb-egyenlö) a paraméterként adott MyList-tel

Parameters:

rhs	- **MyList**

bool MyList::operator<= (const char * *str*)

Összehasonlítja (kisebb-egyenlö) a paraméterként adott karaktertömbbel

Parameters:

str	- karaktertömb

MyList & MyList::operator= (const MyList & *rhs*)

Meglévõ MyList-et írja felül MyList-tel

Parameters:

rhs	- **MyList**

Returns:

meglévö MyList-et adja vissza felülírva

MyList & MyList::operator= (const char * *str*)

Meglévõ MyList-et írja felül karaktertömbbel

Parameters:

str	- karaktertömb

Returns:

meglévö MyList-et adja vissza felülírva

bool MyList::operator== (const MyList & *rhs*)

Összehasonlítja (egyenlö) a paraméterként adott MyList-tel

Parameters:

rhs	- **MyList**

bool MyList::operator== (const char * *str*)

Összehasonlítja (egyenlö) a paraméterként adott karaktertömbbel

Parameters:

str	- karaktertömb

bool MyList::operator> (const MyList & *rhs*)

Összehasonlítja (nagyobb) a paraméterként adott MyList-tel

Parameters:

rhs	- **MyList**

bool MyList::operator> (const char * *str*)

Összehasonlítja (nagyobb) a paraméterként adott karaktertömbbel

Parameters:

str	- karaktertömb

bool MyList::operator>= (const MyList & *rhs*)

Összehasonlítja (nagyobb-egyenlö) a paraméterként adott MyList-tel

Parameters:

rhs	- **MyList**

bool MyList::operator>= (const char * *str*)

Összehasonlítja (nagyobb-egyenlö) a paraméterként adott karaktertömbbel

Parameters:

str	- karaktertömb

char & MyList::operator[] (unsigned int *idx*) const

MyList-et indexelve visszaad egy karaktert Kiszámolja, hogy melyik **MyString** hányadik karaktere a keresett

Parameters:

idx	- index

Returns:

referenciát ad vissza a karakterre

MyList & MyList::urit ()

MyList-böl kitörli a MyString-eket

Üres MyList-et kapunk vissza

The documentation for this class was generated from the following files:

- MyList.h
- MyList.cpp

MyString Class Reference

#include <MyString.h>

Public Member Functions

- **MyString** ()
- **MyString** (const char *s)
- **MyString** (const char *s, unsigned int first)
- **MyString** (const char c)
- **MyString** (const **MyString** &s)
  Másolókonstruktor.
- const char * **getStr** () const
  Visszaadja a MyString-ben tárolt karaktertömb címét
- unsigned int **getSiz** () const
  *Visszaadja, hogy meddig van feltöltve a **MyString**.*
- unsigned int **getRem** () const
  Visszaadja, hogy mennyi üres hely van még a MyString-ben.
- **MyString** & **operator=** (const char c)
- **MyString** & **operator=** (const char *s)
- **MyString** & **operator=** (const **MyString** &s)
  Meglévõ MyString-et felülírja a paraméterként adott MyString-gel.
- char & **operator[]** (unsigned int idx)
- **MyString operator+** (const char c)
- **MyString operator+** (const char *s)
- **MyString operator+** (const **MyString** s)
- **MyString operator*** (int n)

Private Attributes

- char **str** [tarolo_meret+1]
  Tárolt karaktertömb.
- unsigned int **siz**
  Tárolja, hogy a karaktertömb meddig van feltöltve.
- unsigned int **rem**
  Tárolja, hogy a karaktertömbben mennyi üres hely van még

Detailed Description

Fix méretű string, ami karaktertömböt tárol. Méretét a **set.h** fájlból veszi

Constructor & Destructor Documentation

MyString::MyString ()[inline]

Paraméter nélkül hívható konstruktor Üres MyString-et hoz létre

MyString::MyString (const char * *s*)

Paraméterként adott karaktertömbbõl MyString-et csinál

Parameters:

s	- karaktertömb

MyString::MyString (const char * *s*, unsigned int *first*)

Paraméterként adott karaktertömbbõl MyString-et csinál szintén paraméterként adott kezdõponttól kezdve

Parameters:

s	- karaktertömb
first	- kezdõpont indexe

MyString::MyString (const char *c*)

Paraméterként adott karakterbõl MyString-et csinál

Parameters:

c	- karakter

Member Function Documentation

MyString MyString::operator* (int *n*)

Meglévõ **MyString** tartalmát többszörözi meg

Parameters:

n	- szorzó

Returns:

új szorzat MyString-et ad vissza

MyString MyString::operator+ (const char *c*)

Meglévõ MyString-hez füz karaktert

Parameters:

c	- karakter

Returns:

új összeg MyString-et ad vissza

MyString MyString::operator+ (const char * *s*)

Meglévõ MyString-hez füz karaktertömböt

Parameters:

s	- karaktertömb

Returns:

új összeg MyString-et ad vissza

MyString MyString::operator+ (const MyString *s*)

Meglévõ MyString-hez füz MyString-et

Parameters:

s	- **MyString**

Returns:

új összeg MyString-et ad vissza

MyString & MyString::operator= (const char *c*)

Meglévõ MyString-et felülírja a paraméterként adott karakterrel

Parameters:

c	- karakter

MyString & MyString::operator= (const char * *s*)

Meglévõ MyString-et felülírja a paraméterként adott karaktertömbbel

Parameters:

s	- karaktertömb

char & MyString::operator[] (unsigned int *idx*)

MyString-et indexelve visszaadja a benne tárolt karaktertömb egy karakterét

Parameters:

idx	- index

Returns:

referenciát ad vissza a karakterre

The documentation for this class was generated from the following files:

- MyString.h
- MyString.cpp