Házi feladat dokumentáció

Programozás alapjai 2.

Kis-Bogdán Kolos | `N3PTUN`

Választott feladat

Feladatötletek listáról a dinamikus sztringet választottam.

Dinamikus sztring

Készítsen olyan dinamikus sztringet, melyben a sztring karaktereit 20 karakter tárolására alkalmas tárolókból kialakított láncolt listában tárolja! Valósítsa meg az összes értelmes műveletet operátor átdefiniálással (overload), de nem kell ragaszkodni az összes operátor átdefiniálásához! Legyen az osztálynak iterátora is! Legyen képes az objektum perzisztens viselkedésre!

Specifikáljon egy egyszerű tesztfeladatot, amiben fel tudja használni az elkészített adatszerkezetet! A tesztprogramot külön modulként fordított programmal oldja meg! A megoldáshoz ne használjon STL tárolót!

Specifikáció

Működés

A program a leírtak szerint 20 karakter hosszú, dinamikusan foglalt tárolókban tárolja az adatokat, amiket láncolt listába fűz.

Műveletek

Létrehozás

Sztringet lehet létrehozni:

- üresen
- `char`-ból
- `char *`-ból
- másik sztringből

Értékadás | `=`

Ugyanúgy működik, mint a létrehozás, csak felülírja a sztring tartalmát.

Visszatérési értéke: önmaga, a többszörös értékadás lehetővé tétele miatt.

Indexelés | `[]`

Visszaadja az adott indexen lévő karakter referenciáját. Túlindexelés esetén `std::out_of_range` kivételt dob.

Egyenlőség | `==` | `!=`

Összehasonlítja magát egy másik:

- sztringgel
- `char`-ral
- `char *`-gal

Visszatérési értéke `true` megegyeznek (`char`-ral csak egy hosszú sztring egyezhet meg), egyébként `false`.

A `!=` egyértelműen ennek az ellentéte.

Összeadás | `+`

A sztringet összeadja egy másik:

- sztringgel
- `char`-ral
- `char *`-gal

Visszaad egy új sztringet.

Hozzáfűzés | `+=`

A sztringhez hozzáfűz egy:

- sztringet
- `char`-t
- `char *`-t

Visszatérési értéke a jelenlegi sztring referenciája.

Left-shift | krokodil | `<<`

A krokodil operátorral ki lehet írni a sztringet egy `std::ostream`-re. Visszaadja az `std::ostream` referenciát, lehetővé téve az egymás után fűzést.

Right-shift | másfajta krokodil | `>>`

A másfajta krokodil operátorral `std::istream`-ről lehet sztringet beolvasni. Sorvége karakterig (`'\n'` | `'\r'`) vagy `EOF`-ig olvas. Visszaadja az `std::istream` referenciát.

Megszüntetés | destruktor

Felszabadítja az összes dinamikusan foglalt memóriát.

Tagfüggvények

Részsztring képzés | `substr(const size_t from, const size_t length)`

A sztringnek egy megadott részét adja vissza. Kezdő indexnél zárt. Hibás indexelés esetén `std::out_of_range` típusú kivételt dob.

Visszatérési értéke: új sztring, ami csak az adott részt tartalmazza.

C-sztring létrehozása | `c_str(const bool autoFree = true)`

`const char *`-ot csinál a sztringből, amit egy dinamikusan foglalt memóriaterületen tárol. Az `autoFree` `true` értékre állítása esetén a felhasználónak nem kell felszabadítani a memóriaterületet, hanem azt majd a sztring osztály megteszi. `false` értéknél a felszabadítást egyértelműen a felhasználónak kell megtennie.

Visszatérési értéke: `const char *`, ami a sztring tartalmából képzett karaktertömbre mutat, lezáró nulla van a végén.

C-sztring felszabadítása | `free_c_str()`

Felszabadítja a `c_str()` által foglalt memóriát, ha van, a sztring destruktálása előtt.

Hossz lekérdezése | `lenght()`

A sztring jelenlegi hosszának lekérdezése.

Visszatérési értéke: `size_t`

Karakter vagy sztring keresése | `find(const char c)` | `find(const char *s)` | `find(String s)`

Megkeresi a sztringben az első előfordulását az adott keresési feltételnek.

Visszatérési értéke: `size_t`, első előfordulás indexe. Ha nem fordul elő, akkor `std::out_of_range` kivételt dob.

Iterátor

Az osztálynak van iterátora, ami megszokott módon viselkedik.

A sztring elejét a `.begin()`-nel, a végét a `.end()`-el lehet lekérni.

Tesztelés

A tesztelést a Jportán is használt `gtest_lite`-al fogom megvalósítani. A teszt az osztály minden részét, több különböző tesztesettel fogja tesztelni.

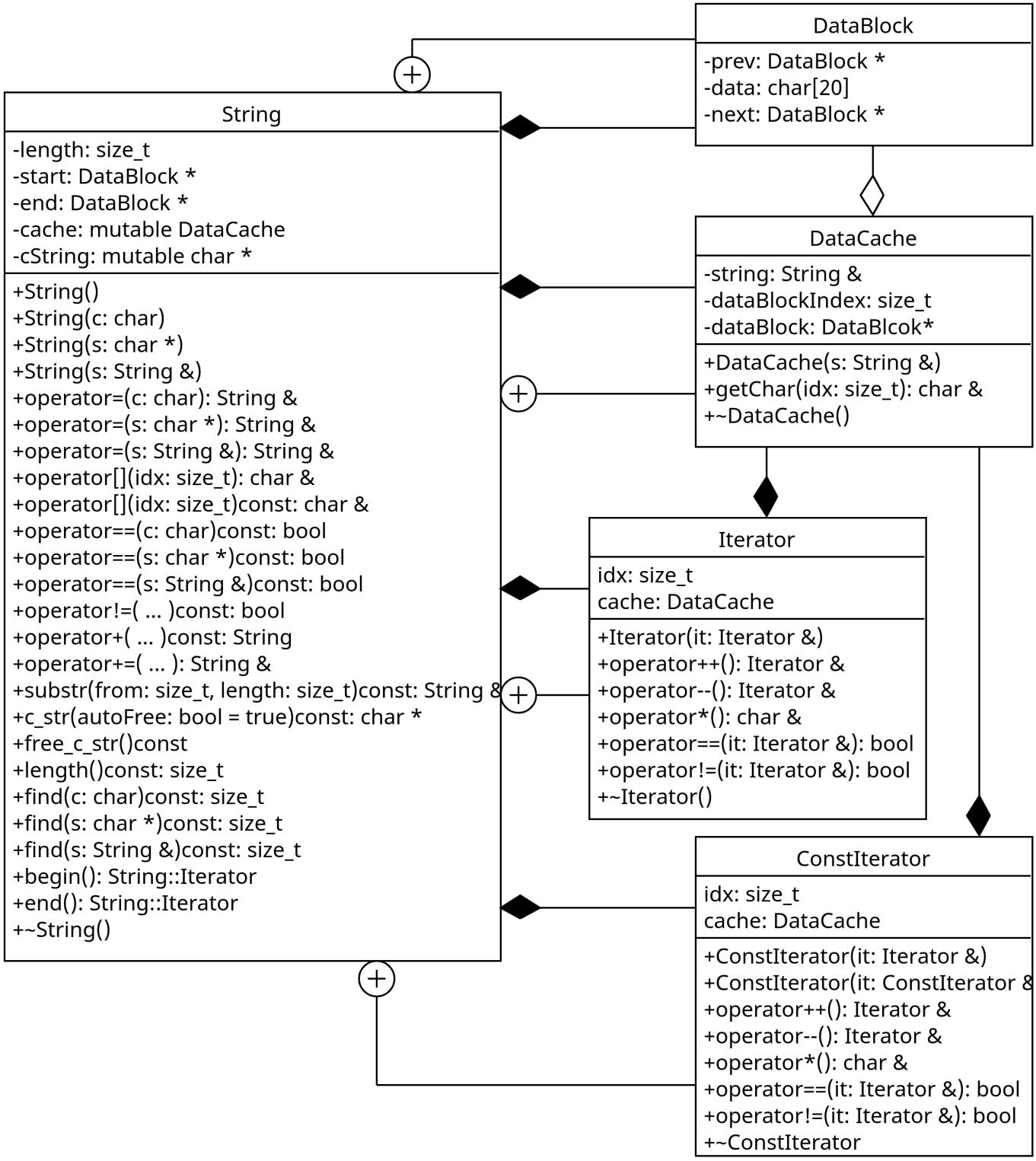
Módosítások a specifikációban

- `substring()` átnevezése `substr()`-re és paraméterek módosítása (`from` és `to` helyett `from` és `length`)
- `cString()` átnevezése `c_str()`-re
- `freeCString()` átnevezése `free_c_str()`-re

- `+` operátor hozzáadása
- `+=` operátor hozzáadása

Terv

UML osztálydiagram



> az `operator!=`, `operator+`, `operator+=` ugyanolyan paraméterekkel hívható, mint az `operator==`

Említésre méltó algoritmusok

Cache

A láncolt lista hozzáférésének gyorsítása a célja. Amikor egy sztringet indexel a felhasználó, az általában egymás utáni karakterekre történik.

Inicializáláskor a sztring első adatblokkjára mutató pointert tárolja el. (A sztringnek mindig van legalább egy adatblokkja)

Karakter indexeléskor két eset lehet:

1. a karakter beleesik a meglévő cache-be (sztring eleje, vége és a cache blokk): Ilyenkor egyszerűen ki lehet olvasni a megfelelő helyről.
2. a karakter nincs benne a cache-ben: Ilyenkor a kért adatblokkhoz a legközelebbi cache-ből el kell jutni a kívánt blokkig, majd kiolvasni a karaktert. Utána a cache-be ez a blokk kerül.

Iterátor

Az iterátorban is van cache, így egy sztringhez egyszerre több iterátorral, több helyen lehet hozzáférni hatékonyan.

`std::istream`-ről beolvasás

A sztring normál állapotában nincs mindig tele az utolsó adatblokk. A beolvasás menete:

1. utolsó adatblokk feltöltése adattal
2. új adatblokk fűzése a sztringhez és ennek feltöltése, amíg van adat

> `std::istream`-ről mindig csak annyi karaktert olvas be, amennyit kell.

Szubsztring képzés

1. teszteli, hogy a `from + length` túlindexeli-e a sztringet, ha igen, kivételt dob
2. új üres sztringet hoz létre
3. karakterenként átmásolja az új sztringbe a kívánt részt

C-sztring képzés

- `autoFree == true`
    1. ha nem `nullptr` a `cString`, akkor azt felszabadítja azt
    2. foglal memóriát `length + 1` karakternek a `cString`-be
    3. blokkonként átmásolja az adatokat a `cString`-be
    4. utolsó blokkból csak annyit másol, amennyi adat van benne
    5. `'\0'`-t tesz a végére
- `autoFree == false`
    1. foglal memóriát `length + 1` karakternek egy ideiglenes változóban
    2. blokkonként átmásolja az adatokat ebbe a változóba
    3. utolsó blokkból csak annyit másol, amennyi adat van benne
    4. `'\0'`-t tesz a végére
    5. `cString`-hez hozzá sem nyúl

> `autoFree == false` esetén ezt a memóriát értelemszerűen a felhasználónak kell felszabadítania.

C-sztring felszabadítás

`free_c_str()` függvénnyel: függvény hívásakor felszabadítja a memóriát és `nullptr`-re állítja a `cString`-et, ha nem volt `nullptr`.

Destruktorral: automatikusan meghívja a `free_c_str()`-t, ha az `autoFree` igaz.

> csak az `autoFree == true`-val végzett `c_str()` képzésnél lehet így felszabadítani a memóriát.

Indexelés

Indexelésnél csak a felső határt kell nézni, mivel a `size_t` nem lehet negatív.