Szepes Nóra – N2WIN2
Szoftver laboratórium 2 NHF dokumentáció
2010/2011/02
2011. május 8.

A feladat szövege

```
Készítsen GENERIKUS tömböt! Valósítsa meg a shell rendezést, mint az
osztály tagfüggvényét.
A tömb rendezéséhez használjon predikátumobjektumot.

Valósítsa meg az összes értelmes műveletet operátor átdefiniálással
(overload), de nem kell ragaszkodni mindenáron az operátorok
átdefiniálásához!
Amennyiben lehetséges használjon iterátort!

Demonstrálja a működést külön modulként fordított tesztprogrammal!
A programmal mutassa be a generikus szerkezet használatát több egyszerű
adathalmazon, amit fájlból olvas be!
A megoldáshoz NE használjon STL tárolót vagy algoritmust!
```

Pontosított specifikáció

Készítsen GENERIKUS tömböt!

- Létrehoztam egy osztályt, mely generikus tömböt (T *t), és annak méretét tárolja.
- A konstruktorral megadott méretű tömböt dinamikusan kell lefoglalni, és a program végén azt megsemmisíteni azt a destruktorral.

Valósítsa meg a shell rendezést, mint az osztály tagfüggvényét.

- Létrehoztam két shell függvényt, egyiket char*-ra, másikat pedig számokra, mint double és int típusok. Bemenetként meg kell adni egy számot, ami ha 0nál kisebb, akkor csökkenőbe rendez, különben pedig növekvőbe.
- Predikátum objektumként (vagyis külső függvény használataként) írtam egy compare függvéyt, és azzal döntöm el, hogy melyik szám nagyobb/kisebb. Char*nál a már megírt strcmp-t használtam.

Valósítsa meg az összes értelmes műveletet operátor átdefiniálással (overload), de nem kell ragaszkodni mindenáron az operátorok átdefiniálásához! Amennyiben lehetséges használjon iterátort!

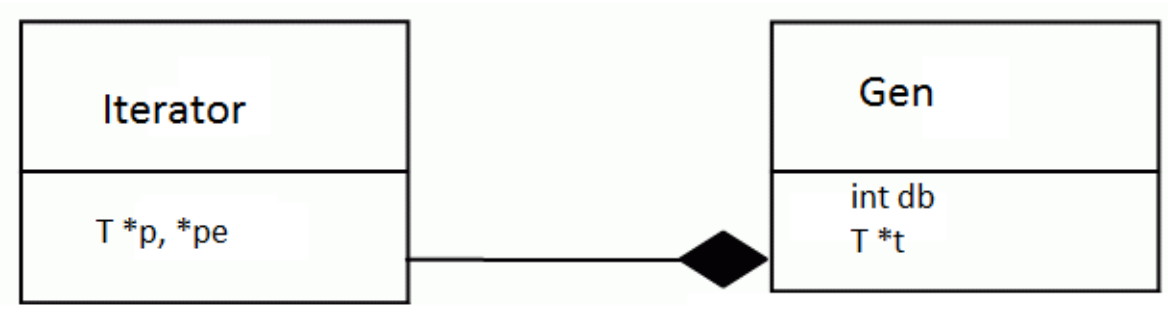
- Először létrehoztam egy iterátor osztályt a generikus osztályon belül, majd ennek a segítségével iterátorokkal tudom végigjárni a tömböket, amiket a main-ben hozok létre. Valamint az iterátorok segítségével megcsináltam az operátor overloadokat.

Operátor overload

- Létrehoztam 12 operátort, melyekkel összeszorozni, összeadni, és kivonni tudok tömböket, vagy egy tömbhöz hozzáadni, tömbből kivonni illetve azt megszorozni egy számmal.
- Ezen kívül létrehoztam még egy indexoperátort ([]), mellyel lekérdezhetem a tömb i-edik elemét, egy értékadó (=) és egy egyenlőség vizsgáló operátort (==).

Szepes Nóra – N2WIN2
Szoftver laboratórium 2 NHF dokumentáció
2010/2011/02
2011. május 8.

Objektummodell



Interfész leírás

Osztályok

class	iterator

Publikus tagfüggvények

	Gen (int darab=6)
	Gen (Gen const &e)
iterator	**begin** ()
iterator	**end** ()
int	**Getdb** ()
int	**compare** (T a, T b)
void	**shell** (int n)
void	**strshell** (int n)
void	**Print** ()
T &	**operator[]** (int i)
int	**operator==** (Gen &e)
Gen &	**operator=** (const Gen &a)
Gen	**operator+** (Gen &d)
Gen &	**operator+=** (Gen &d)
Gen	**operator+** (double elem)
Gen &	**operator+=** (double elem)
Gen	**operator*** (Gen &d)
Gen &	**operator*=** (Gen &d)
Gen	**operator*** (double c)
Gen &	**operator*=** (double c)
Gen	**operator-** (Gen &d)
Gen &	**operator-=** (Gen &d)
Gen	**operator-** (double elem)
Gen &	**operator-=** (double elem)
void	**StrLenAvg** ()
void	**StrLenMax** ()
void	**StrLenMin** ()
void	**Modusz** ()
void	**Median** ()
T	**GetMax** ()
T	**GetMin** ()
T	**GetSum** ()
T	**GetSzorzat** ()
double	**GetAvg** ()

Gen osztály:

A program fő osztálya a Gen osztály, mely tartalmaz egy T típusú t nevű pointert (T *t), mellyel meghatározhatjuk, hol van a memóriában a tömb számára lefoglalt dinamikus rész. A darabszámot pedig egy int db nevű változó tárolja. Ezen kívül a fő tagfüggvények is itt deklaráltam (gen.hpp-ben és a gen.cpp-ben fejtettem őket ki), valamint az iterátor osztályt is.

Iterátor osztály:

Tartalmaz két T típusú pointert, az első és az utolsó elemre (T *p, *pe). Itt létrehoztam többek között léptető függvényeket, melyek segítségével tudtam végigjárni a tömböt, valamint hogy össze tudjam hasonlítani, hogy az adott elem már az utolsó utáni-e (vagyis már nincs benne a tömbben.

Ehhez hozzátartozik még a Gen osztályban definiált begin és end függvények, melyekkel beállíthattam az első és az utolsó utáni elemre az iterátort.

Publikus tagfüggvények

	iterator (Gen &a, int ix=0, int siz=0)
iterator &	**operator++** ()
iterator	**operator++** (int)
bool	**operator!=** (const iterator &i)
T &	**operator*** ()
T *	**operator->** ()
T	**iter** ()

Szepes Nóra – N2WIN2
Szoftver laboratórium 2 NHF dokumentáció
2010/2011/02
2011. május 8.

A megoldás rövid váza

A megoldás első lépéseként elterveztem, hogy milyen osztályokra lesz szükségem, valamint hogy milyen főbb függvényeket kell létrehoznom. Elsőként létrehoztam a Gen osztályt, mely a fő része a programnak, hisz ez az osztály tulajdonképpen maga a generikus tömb. Három fő típusban kezdtem el gondolkodni, ezek az int, double és char* típusok. Inline függvénynek csak a konstruktor, destruktor, copy konstruktor valamint az iterátoros függvényeket hagytam meg.

A többi függvényt a gen.cpp nevű fájlban fejtettem ki. Először az úgynevezett szám típusokra írtam meg a függvényeket. Ilyen a 12 operátor függvény (*, *=, +, +=, -, -=). Külön létrehoztam 2-2-2 szorzás, összeadás és kivonás függvényt, külön verziónak tekintettem azokat, amelyeknél egy másik tömbbel végzek műveletet, vagy egy T számmal, illetve ahol az eredeti tömböt írom felül vagy ahol egy új tömböt adok vissza eredményként. Az operátorok a kisebb elemű tömb utolsó eleméig futnak. Majd elkészítettem az indexoperátort, az értékadó és az egyenlőség operátorral együtt.

Következő lépésként megírtam az értékeket visszaadó függvényeket, ilyenek az átlagot, darabszámot vissza, móduszt, mediánt, minimumot, maximumot, összeget, szorzatot visszaadó függvények számoknál, míg char*oknál az átlag szóhossz, leghosszabb és legrövidebb szó függvény.

Az iterátor osztályhoz létrehoztam két függvényt, melyekkel az első és az utolsó utáni elemre tudtam állítani az iterátorokat, illetve magát az iterátor osztályt. Az iterátor osztályon belül többek között léptető, és összehasonlító függvényeket hoztam létre.

És a program úgynevezett másik fő része a shell függvények. Mint már említettem, kétféle shell van, egy számokra és egy karakterekre. Paraméterként bekérnek egy számot, ha az kisebb mint 0, akkor csökkenőbe rendezi a tömböt, különben pedig növekvőbe. Predikátum objektumként én a compare függvényre gondoltam, charnál van egy beépítet strcmp függvény, míg számokhoz létrehoztam a compare függvényt.

Main és a tesztesetek leírása

Első részként az operátorokat teszteltem, majd a számokra írt külön függvényeket. Itt megadott számokkal dolgoztam, mert a számokkal való műveletnél jobban látszanak, hogy miként is működnek. Ezek után jön a shell tesztelés, melyekhez double tömböket olvasok be fájlból. A fájlokat a consoleon keresztül tudja megadni a felhasználó. Ha a fájl nem található, akkor ez a rész kimarad. Alapból két fájlt kérek be, hogy mind növekvő, mind csökkenő rendezést bemutassak, természetesen különböző tömbökön. A char* shell rendezéséhez szintén beolvasok 1 tömböt, fájlból, majd ezek után tesztelem a shelleket, mind növekvően, mind csökkenően, majd az eddig még nem meghívott függvényeket.

Három tesztesetet csináltam Cportán. Az első tesztesetnél a double1.dat, double2.dat és string.dat fájlokat használom, a második tesztesetnél a double3.dat, double4.dat és string2.dat fájlokat. Harmadik tesztesetként csak egy entert hagytam bemenetként, innen látszik, hogy nem létező fájlok esetén olyan része nem fut le a programnak, melyhez beolvasott tömbre van szükség.