Házi feladat

Szoftver laboratórium 2.

Tóth András

(O8POUA)

Generikus duplán láncolt négy elágazású fa

Tartalom

1. Feladat 2

2. Pontosított feladatspecifikáció 3

Point: 3

QuadTree: 3

QuadTree::iterator: 3

QuadTreeNode: 3

3. Terv 4

Point adattagjai és tagfüggvényei 4

QoudTree adattagjai és tagfüggvényei 4

QuadTree::iterator adattagja és tagfüggvényei 4

QuadTreeNode adattagjai és tagfüggvényei 5

Iterátor működése (négyfa postorder bejárása): 5

Osztálydiagram: 6

# OSX:Users:tothandras:Dropbox:Mernokinformatika:Szoftlab2:Szoftlab2NHFquadtree:NHFQuadTree:Others:QuadTreeLogo.png1. Feladat

Tóth András (O8POUA) részére:

Készítsen GENERIKUS duplán láncolt 4 elágazású fát (quad-tree)! Valósítsa meg az összes értelmes műveletet operátor átdefiniálással (overload), de nem kell ragaszkodni az összes operátor 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, és egy olyan saját osztályon, amely dinamikus adatot tartalmaz A megoldáshoz NE használjon STL tárolót vagy algoritmust! A tesztprogramot úgy specifikálja, hogy az parancssoros batch alkalmazásként (is) működjön, azaz a szabványos bemenetről olvasson, és a szabványos kimenetre, és/vagy a hibakimenetre írjon! Lehetősége van grafikus, vagy kvázi grafikus interaktív felhasználói felület kialakítására is, de fontos, hogy a Cporta rendszerbe olyan változatot töltsön fel, ami ezt nem használja! Amennyiben a feladat teszteléséhez fájlból, vagy fájlokból kell input adatot olvasnia, úgy a fájl neve \*.dat alakú legyen!

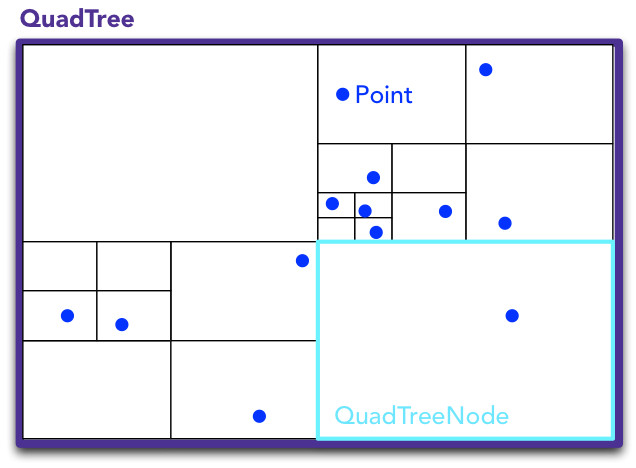
# 2. Pontosított feladatspecifikáció

A feladat egy generikus duplán láncolt négy elágazású fa (továbbiakban négyfa, angolul quadtree) készítése. A négyfa egy olyan fa struktúra, amiben minden csúcsnak pontosan négy gyereke van. A négyfát leggyakrabban két-dimenziós tér felbontására használják oly módon, hogy a tér rekurzívan felbontható kisebb negyedekre. Ezek a területek leggyakrabban négyzetek, vagy téglalapok. A feladat nem specifikálja, hogy milyen módon lehessen használni a négyfát, ezért az előbbiekben leírt két-dimenziós tér felbontására lesz használható. A feladat specifikációja arra sem tér ki, hogy milyen objektumokkal valósítsam meg a fát. A négyfát ezért ezekkel az objektumokkal valósítom meg:

## Point:

Az adatok pontokban tárolhatók el. A pontnak két koodrinátája van (x és y) és egy változója, amelyben a generikus adat tárolható.

## QuadTree:

A felhasználó ilyen QuadTree objektumokat hozhat létre. Az objektum elzárja a külvilág elől a fa felépítését.

A QuadTree tagfüggvényei:

* beszúrás
* mélység megszámolása
* elemek megszámolása
* keresés adott pont szerint
* keresés adat szerint

## QuadTree::iterator:

A QuadTree osztály iterátora. Segítségével a négyfa csúcsait járhatjuk be.

## QuadTreeNode:

A QuadTree osztály ilyen objektumokból építi fel a négyfát. Az osztály el van rejtve a külvilág elől.

# 3. Terv

A generikus négy elágazású fa az alábbi osztályokból épül fel:

## Point adattagjai és tagfüggvényei

adattagjai:

* double x
* double y
* T data

(x, y vízszintes és függőleges koordináták, data a generikus adat)

tagfüggvényei:

* konstruktor
* destruktor
* operator==
* operator!=
* getData (visszatér a tárolt generikus adattal)

## QoudTree adattagjai és tagfüggvényei

adattagjai:

* root (fa gyökerére mutató pointer)
* iterator

tagfüggvényei:

* konstruktor
* destruktor
* insert (pont beszúrása)
* depth (mélység visszaadása)
* countNodes (csúcsom megszámolása)
* find (pont keresése)
* find (adat keresése)
* begin() (az első elemre mutató iterator)
* end() (az utolsó utáni elemre mutató iterator)

## QuadTree::iterator adattagja és tagfüggvényei

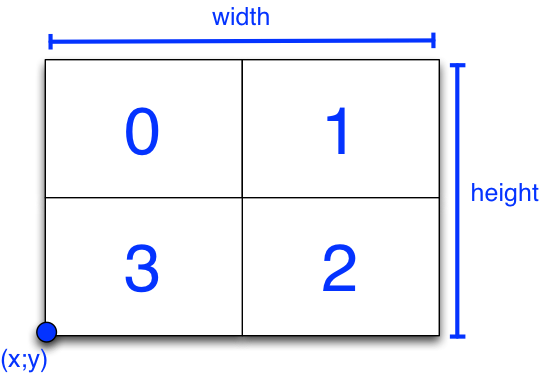
adattagjai:

* node (jelenlegi elem)

tagfüggvényei:

* konstruktor
* operator\*
* operator->
* operator++ (prefix és postfix)
* operator==
* operator!=

## QuadTreeNode adattagjai és tagfüggvényei

adattagjai:

* parent (szülőre mutató pointer)
* children[4] (gyerekekre mutató pointer)
* point (tárolt pontok dinamikus tömbje)
* number\_of\_points (pontok száma)
* x, y, width, height (területre jellmző adatok)
* level (fában lévő szintje)
* MAX\_LEVEL (szintek maximális száma)

tagfüggvényei:

* konstruktor
* destruktor
* split (gyerekek létrehozása megfelelő adattagokkal)
* insert (pont beszúrása)
* hasData (igaz, ha van pont/adat a csúcsban)
* getLevel
* isLeaf (igaz, ha levél)

## Iterátor működése (négyfa postorder bejárása):

1. A legbaloldalibb levéltől indul.

2.a A szomszédos csúcsra a következő elem, ha az levél.

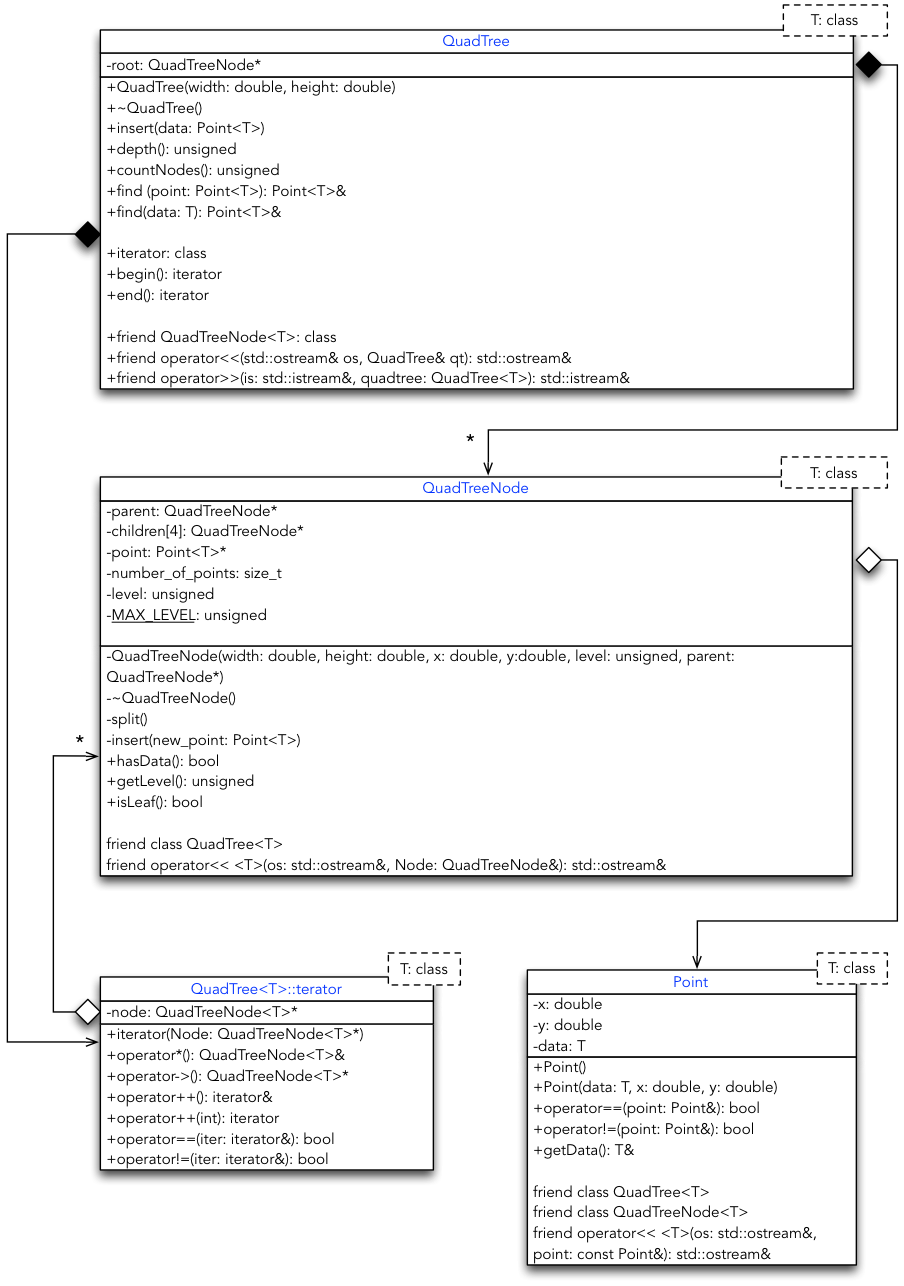
2.b A szomszédos csúcs legbaloldalibb levelél a következő elem, ha létezik.

2.c Ha a negyedik csúcsnál vagyunk, akkor a következő elem a szülő.

3. Az utolsó elem a fa gyökere.

Iterátor bejárásának szemléltetése ábrán:

## OSX:Users:tothandras:Dropbox:Mernokinformatika:Szoftlab2:Szoftlab2NHFquadtree:NHFQuadTree:QuadTree_iterator.pngOsztálydiagram:



## Tesztprogram

# 4. Megvalósítás