Semestrálna práca 1

**Algoritmy a údajové štruktúry 2**

Autor: Bc. David Kučera

Štud.skup.: 5ZIS12

Obsah

[Popis semestrálnej práce 3](#_Toc181373824)

[Dátová štruktúra 3](#_Toc181373825)

[Nákres dátovej štruktúry 3](#_Toc181373826)

[Priemerné časové zložitosti implementovanej dátovej štruktúry 4](#_Toc181373827)

[Výpočtová zložitosť operácií 4](#_Toc181373828)

[Architektúra práce 4](#_Toc181373829)

[UML diagram tried modulu DataStructures 5](#_Toc181373830)

[UML diagram tried modulu GeoLib 6](#_Toc181373831)

[Opis implementácie operácií 6](#_Toc181373832)

[Hľadanie 6](#_Toc181373833)

[Vkladanie 6](#_Toc181373834)

[Update 6](#_Toc181373835)

[Mazanie 6](#_Toc181373836)

[Návrh riešenia práce so súbormi 9](#_Toc181373837)

[Ukážka uloženého súboru z aplikácie 10](#_Toc181373838)

# Popis semestrálnej práce

Cieľom semestrálnej práce bolo vytvoriť **demonštračnú verziu** softvéru pre informačný systém záznamov o nehnuteľnostiach, pričom bolo požadované, aby tieto dáta boli uložené v operačnej pamäti.

V tejto verzii je umožnený výpis všetkých evidovaných údajov na obrazovku a nasledovné **operácie**:

1. vyhľadanie nehnuteľností,
2. vyhľadanie parciel,
3. vyhľadanie všetkých objektov,
4. pridanie nehnuteľnosti,
5. pridanie parcely,
6. editácia nehnuteľnosti,
7. editácia parcely,
8. vyradenie nehnuteľnosti a
9. vyradenie parcely.

# Dátová štruktúra

Ako dátová štruktúra pre uchovávanie údajov bol zvolený **k-dimenzionálny strom** (k-d strom), ktorý podporuje viacrozmerné bodové vyhľadávanie. Ako kľúče boli zvolené GPS pozície objektov a v jednotlivých dimenziách sa vetví podľa šírky a dĺžky, teda strom má 2 dimenzie.

## Nákres dátovej štruktúry

Aplikácia obsahuje 3 k-d stromy – jeden obsahujúci referencie na parcely, jeden na nehnuteľnosti a tretí strom na oba typy štruktúr. V strome geo-objektov sú **referencie** na **rovnakú** **inštanciu** parcely, ako v strome parciel a rovnako tak v prípade nehnuteľností.

K-d strom **parciel** K-d strom **nehnuteľností**

Obrázok, na ktorom je kruh

Automaticky generovaný popis Obrázok, na ktorom je kruh

Automaticky generovaný popis

(počet **p**) (počet **n**)

K-d strom **geo-objektov**

Obrázok, na ktorom je kruh

Automaticky generovaný popis

(počet **p+n**)

## Priemerné časové zložitosti implementovanej dátovej štruktúry

|  |  |  |  |
| --- | --- | --- | --- |
| **Operácia** | K-d strom **parciel** | K-d strom **nehnuteľností** | K-d strom **geo-objektov** |
| Vlož | O(log2 p) | O(log2 n) | O(log2 (p+n)) |
| Nájdi | O(log2 p) | O(log2 n) | O(log2 (p+n)) |
| Vymaž | O(log2 p) | O(log2 n) | O(log2 (p+n)) |

## Výpočtová zložitosť operácií

|  |  |  |
| --- | --- | --- |
| Č. | **Operácia** | Výpočtová zložitosť |
| 1. | Vyhľadanie nehnuteľností | O(log2 n) |
| 2. | Vyhľadanie parciel | O(log2 p) |
| 3. | Vyhľadanie všetkých objektov | O(log2 p+n) |
| 4. | Pridanie nehnuteľnosti | O(log2 n) |
| 5. | Pridanie parcely | O(log2 p) |
| 6. | Editácia nehnuteľnosti | O(log2 n) |
| 7. | Editácia parcely | O(log2 p) |
| 8. | Vyradenie nehnuteľnosti | O(log2 n) |
| 9. | Vyradenie parcely | O(log2 p) |

*p, n – počet parciel, počet nehnuteľností*

# Architektúra práce

Aplikácia je rozdelená do 3 hlavných programových modulov.

Prvým najnižším modulom sú **DataStructures**, kde je implementácia všeobecného k-d stromu a dát, ktoré sa v aplikácií využívajú, ako napr. nehnuteľnosť, parcela, GPS.

Ďalším modulom je modul **GeoLib**, ktorý obsahuje jadro aplikácie. Využíva modul DataStructures a vykonáva nad ním operácie pre prácu s údajmi a štruktúrami. V rámci tohto modulu je implementovaný aj generátor dát.

Posledným najvyšším modulom je **GeoViewer**, ktorý obsahuje GUI rozhranie pre prácu s modulom **GeoLib**.



## UML diagram tried modulu DataStructures

Obrázok, na ktorom je text, snímka obrazovky, písmo, diagram

Automaticky generovaný popis

Abstraktnú vrstvu tvoria triedy AbstractNode a AbstractTree, ktoré majú dve generiká a to kľúč a dáta. Priama implementácia nášho stromu je v triede KdTreeNode a KdTree, ktoré dedia z abstraktnej vrstvy. Kľúče, ktoré sa vkladajú do stromu musia mať naimplementované rozhranie IComparable. Do stromu vkladáme v tejto práci objekty GeoObjekt, Parcela, Nehnutelnost. Tie majú ako kľúč objekt GpsPos.

## UML diagram tried modulu GeoLib

Obrázok, na ktorom je text, snímka obrazovky, písmo, rovnobežný

Automaticky generovaný popis

Rozhranie IKatSys definuje operácie, ktoré implementuje trieda KatSys. Tieto operácie tvoria jadro samotnej aplikácie a využívajú sa v GUI. Trieda DataGenerator sa používa na generovanie náhodných dát a hodnôt v triede KatSys.

## Opis implementácie operácií

### Hľadanie

Vstupom je **kľúč**, ktorý chceme vyhľadať. Hľadanie prebieha rovnako ako vo vyhľadávacom strome. Používa sa komparátor, vďaka ktorému vieme, kam nasledovať v prehliadke vrcholov. V prípade, že sa v strome nachádza viac prvkov s rovnakým kľúčom, vrátia sa všetky prvky.

### Vkladanie

Vstupom je **kľúč** a **dáta**, ktoré spolu reprezentujú konkrétny vrchol stromu. Ak ešte strom neobsahuje koreň, nový prvok sa vloží ako koreň. Ak však strom už koreň obsahuje je potrebné nájsť jeho správne miesto a popri tom dodržať pravidlá vyhľadávacieho stromu, teda ako pokračovať v prehliadke stromu – pomocou komparátora. V prípade, že sa vkladá duplicitný kľúč, pokračuje sa v prehliadke **doľava**.

### Update

Táto operácia sa vykonáva o úroveň vyššie, v module KatSys a to nasledovným spôsobom. Ak sa v rozhraní **nezmení** **kľúčový** atribút, tak sa jednoducho **prepíšu** zmenené dáta na nové. Ak sa však zmenili kľúčové atribúty (v našom prípade GPS), tak je nutné daný prvok zo stromu **vymazať** a následne **vložiť** s novými údajmi.

### Mazanie

Vstupom je **kľúč** a **dáta**, ktoré **jednoznačne** reprezentujú konkrétny prvok v strome. Nájdeme daný prvok.



Mazanie prvku zo stromu prebieha nasledovne. Ak je prvok **listom** – teda nemá žiadnych synov, jednoducho sa **odstráni**.



Ak však obsahuje aspoň 1 syna musím mu nájsť nasledovníka. Ten sa vyberá nasledovne. Vždy sa snažím hľadať nasledovníka najskôr z **ľavého** stromu(max) – v tomto prípade nemusím sledovať možné duplikáty a teda len jednoducho prvky vymieňam **cyklicky**.



Avšak, ak sa nepodarí nájsť nasledovníka v ľavom podstrome (syn vľavo neexistuje), je nutné prehľadať **pravý** podstrom, v ňom nájsť nasledovníka(min), ale aj overiť, či sa v ňom **nenachádzajú** **duplikáty**. Ak sa nenachádzajú – pokračujem cyklicky – ako v predošlom prípade. Ak však v pravom podstrome sa **nachádzajú** duplikáty podľa kľúča v danej dimenzii, celý pravý podstrom sa zozbiera, **vymaže** a následne opäť **vloží** do stromu – teraz už na správne miesta – teda nenastane situácia, že by sme nevedeli pristúpiť k daným duplicitným prvkom.



## Návrh riešenia práce so súbormi

Pre zabezpečenie uchovania dát bolo potrebné navrhnúť a implementovať vlastné ukladanie a znovu načítanie dát z a do aplikácie. Nakoľko sa nám jedná o čo **najmenšiu** **veľkosť** súboru, je nutné, aby sa ukladali len potrebné dáta bez akýchkoľvek zbytočností.

Preto sa z aplikácie ukladajú všetky nehnuteľnosti a parcely do **jediného súboru** typu csv. Ako rozdeľovač sa používa bodkočiarka.

### Ukážka uloženého súboru z aplikácie

Nižšie je možné si všimnúť hlavičku súboru a aj pár ukážkových dát. Čo riadok, to jedna hodnota – buď nehnuteľnosť, alebo parcela.

