Semestrálna práca

AUDS 1

Ratkovský Adam

2021/2022

Obsah

Návrh údajových štruktúr

Územné jednotky

Pre uloženie obcí, okresov a krajov som použil **Sorted Sequenced Table**. Je najvhodnejšia, pretože dáta, ktoré sme dostali majú jedinečný názov. Tento názov sa použije ako *kľúč*. Niektoré názvy obcí sa na Slovensku opakujú. To je vyriešené jedinečným názvom „*názov_obce názov_okresu*“ – príklad – „Sása Okres Zvolen“. Takýto názov majú len spomínané duplicitné obce.

Vek a vzdelanie

Dáta vek a vzdelanie sú uložené v **Poli (Array)**. Použitý je kvôli priamemu prístupu na daný index. Pre *vek* sú to 2 polia, pole pre mužov a pre ženy, obidva o veľkosti 101. Pre *vzdelanie* uchovávame 1 pole o veľkosti 8. *Vzdelanie* a *vek* majú svoju vlastnú triedu s týmito poľami. Zároveň obsahujú aj getery a seteri nad poľami podľa potreby.

Uloženie dát

Každá územná jednotka má svoju triedu. Poznáme triedy štát, kraj, okres, obec (pre potreby dokumentácie preložené do slovenčiny). Všetky sú potomkom triedy Územná jednotka. V tej sa nachádzajú atribúty tried **vek** a **vzdelanie**. Ďalej sa tu nachádza atribút objektu **ÚJ**, ktorý odkazuje na nadradenú ÚJ. Atribúty **názov**, **code** a **note**, tak isto aj príslušné geteri. Tieto atribúty zdedia dané územné jednotky.

Dáta sú prepojené nasledovne. Každá trieda má **Sorted Sequenced Table** (SST) smerníkov na územné jednotky.

Štát

- SST všetkých územných jednotiek
- SST svojich krajov (keďže sa jedná o štát, tak všetkých Slovenských krajov)
- SST všetkých obcí
- SST všetkých okresov

Kraj

- SST svojich obcí
- SST svojich krajov

Okres

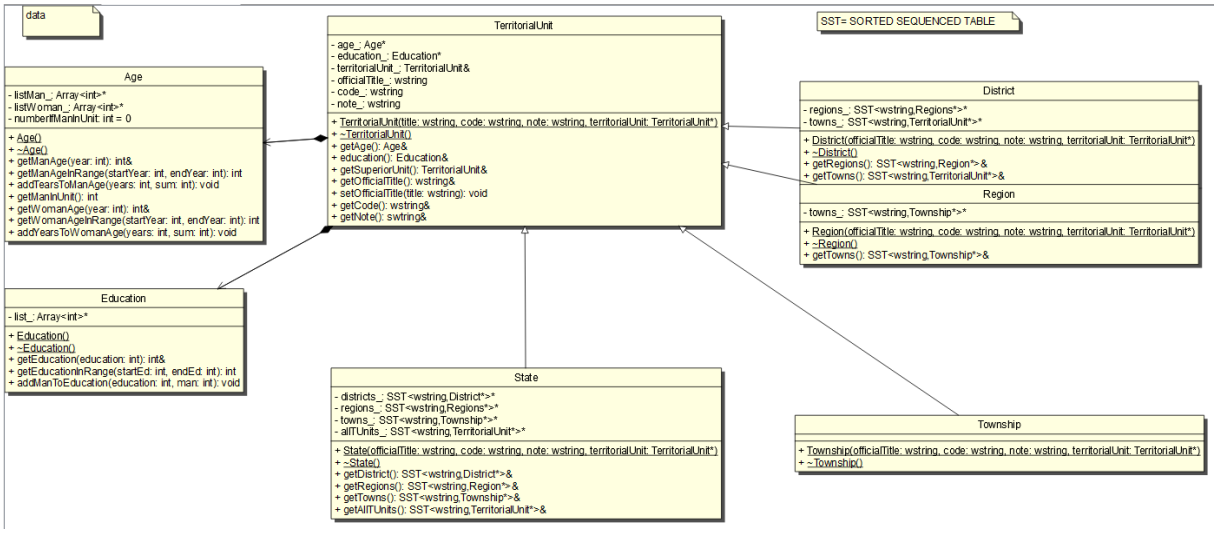
- SST svojich obcí

Takto uchované údaje nám umožňujú vyfiltrovať dáta a vykonávať operácia pomocou kritérií v rýchlych časoch.

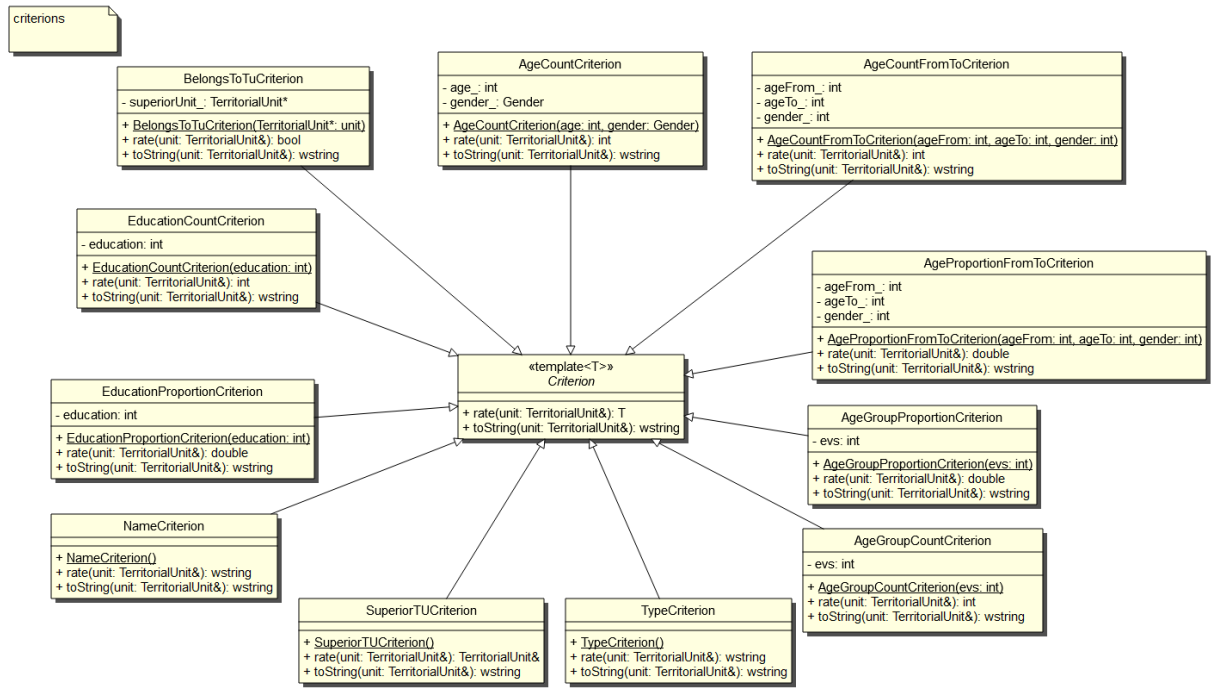
UML diagramy

Diagramy sa nachádzajú aj v prílohe dokumentácie.

Uloženia dát:



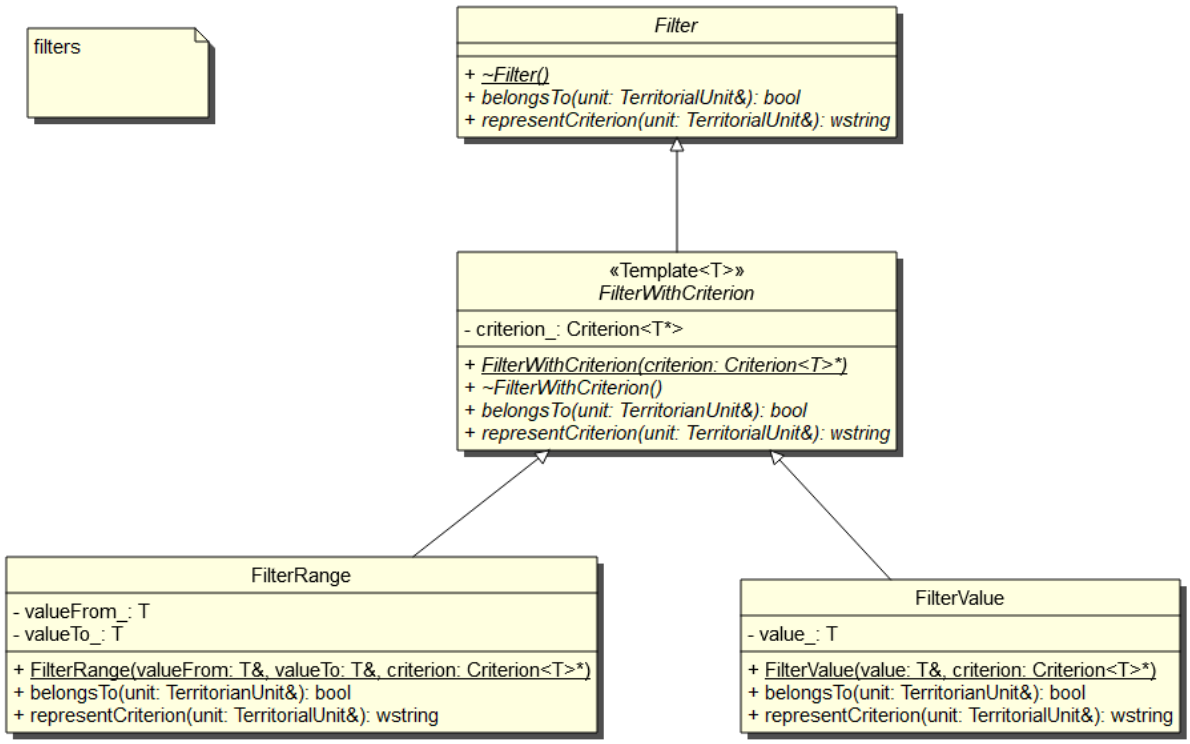
Kritériá:



Kritéria majú spoločného predka, Abstraktnú triedu **Criterion**, s parametrom **T**, tento parameter je návratový typ daného kritéria. Obsahuje virtuálne metódy rate, ohodnoť. Vracia hodnotu **T** pre danú ÚJ. Metódu toString, zavolá metódu rate a vypíše reťazec reprezentujúci kritérium. Samotné kritéria obsahujú implementácie týchto metód. V niektorých kritériách máme parametrický konštruktor. Napríklad kritérium **AgeCountFromTo** prijíma vek, od koľko do koľko, a pohlavie. Metóda rate potom

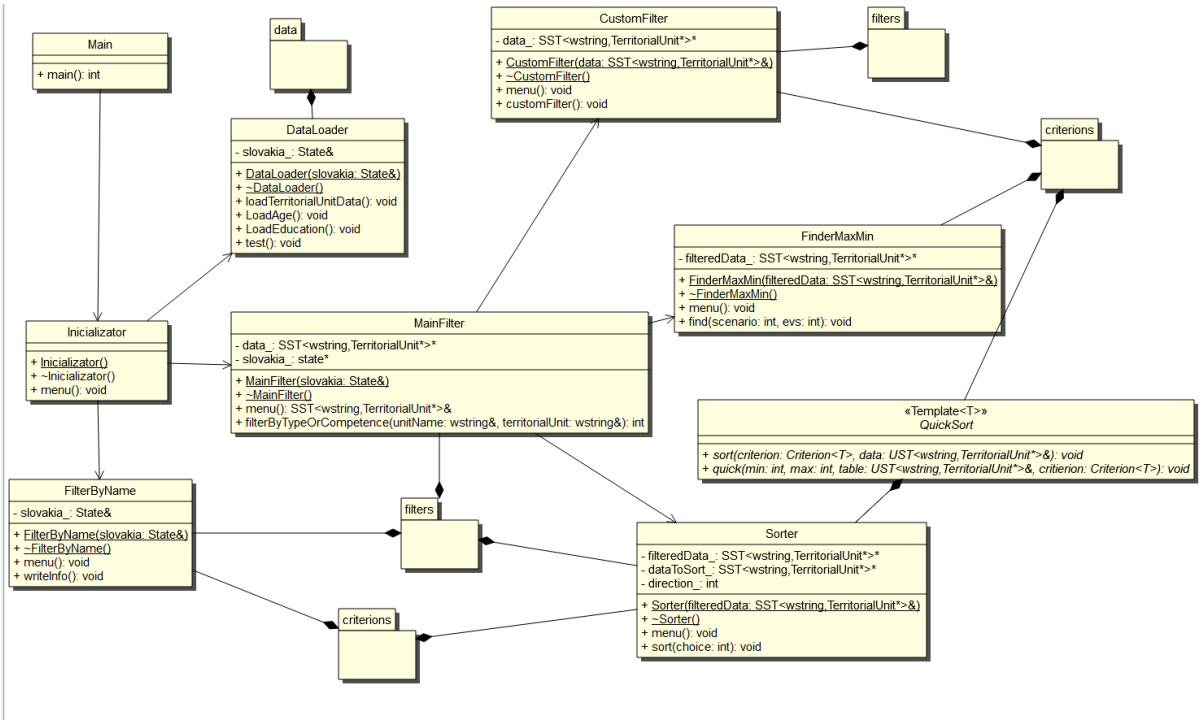
príjme ÚJ a vráti vypočítanú hodnotu. Metóda toString vypíše informáciu koľko obyvateľov akého pohlavia má sa nachádza v zadanom veku od do.

Filtre:



FilterWithCriterion je spoločný predok tried **FilterRange** a **FilterValue**. Obsahuje virtuálnu metódu belongsTo, ktorá volá metódu kritériá rate. Vracia boolean, či územná jednotka spĺňa alebo nespĺňa zadané hodnoty. Virtuálnu metódu representCriterion, ktorá volá metódu KRITÉRIA toString. Pomocou takejto implementácie vieme vypísanie hodnoty ÚJ pre užívateľom zadané filtre. **FilterRange** filtruje územné jednotky podľa vlastnosti v zadanom rozsahu. **FilterValue** filtruje územné jednotky, ktorých vlastnosť je presná zadanej hodnote. Tieto hodnoty posielame do konštruktora spolu s KRITÉRIOM. Trieda **Filter**, ktorá je predok triedy **FilterWithCriterion** je pomocná pre potreby vytvorenia zloženého filtra v ArrayListe, keďže nevieme parameter **T** pri inicializovaní ArrayListu.

Inicializácia:



- Balíčky **criterions** a **filters** sú v diagrame zobrazené 2 krát. Je to z dôvodu lepšej prehľadnosti diagramu a obidve reprezentujú ten istý balíček.

V triede **Main** vytvoríme **Inicializator**, ten obsahuje hlavné menu. Načíta dáta pomocou triedy **DataLoader**, potom podľa voľby používateľa zavolá inštancie tried **MainFilter** alebo **FilterByName**. **MainFilter** vyfiltruje dáta podľa typu a príslušnosti k ÚJ. Tie ďalej pošle podľa voľby do inštancií tried **CustomFilter, FinderMaxMin** alebo **Sorter. CustomFilter** obsahuje zložené filtre. **FinderMaxMin** vyhľadáva ÚJ s najväčšou / najmenšou vekovou skupinou. Trieda **Sorter**, triedi dáta. Trieda **FilterByName** hľadá ÚJ a následne o nej vypisuje informácie.

Popis výpočtov

Načítanie dát

Územné jednotky

- Kraje sa načítajú ako prvé a odkazy na inštancie sa pridajú do SST všetkých ÚJ a SST krajov inštancie triedy štát. Nenačítajú sa len tie, ktoré sú zahraničné. Keďže ÚJ sú na seba odkazované pomocou atribútov **note** a **code**, zahraničné ÚJ sa už nebudú načítavať. Keďže nebudú prislúchať s existujúcimi nadradenými ÚJ. Táto zložitosť je **O(N),** kde **N** je počet všetkých krajov.
- Odkazy na inštancie okresov sa pridajú do SST všetkých okresov a SST všetkých ÚJ v inštancií triedy štát. Cez cyklus for each sa prechádza SST krajov v inštancií triedy štát a hľadá kraj, do ktorého patrí. Toto hľadanie sa vykonáva podľa atribútu **note**. Táto zložitosť je nanajvýš **O(N*M)**, kde **N** je počet všetkých krajov a M je počet všetkých okresov.
- Odkazy na inštancie obcí sa pridajú do SST všetkých obcí, SST všetkých ÚJ v inštancií triedy štát. Ďalej sa cez cyklus for each hľadá, do ktorého kraja z SST krajov v inštancií triedy štát. Pre každý kraj sa prechádza SST okresov a zisťuje sa, kam daná obec patrí. Ak obec patrí do daného okresu, tak sa pridá do SST okresu a kraju. Zložitosť pridávania obcí je nanajvýš **O(N*M*O),** kde **N** je počet krajov, **M** je počet okresov a **O** je počet obcí.

Vek a vzdelanie

Pre vek a vzdelanie musíme v cykle for each prejsť SST tabuľky odkazujúce na potomkov, až pokým sa nedostaneme k obci. Toto porovnanie robíme cez atribúty **note** a **code**. Potom daným inštanciám pridávame vek a vzdelanie podľa danej zistenej položky (muž, žena, stredné vzdelanie, . .). Napríklad prečítame vek obce Pliešovce, vek pridáme štátu, kraju, okresu a nakoniec obci. Zložitosť týchto operácií je pri najhoršom prípade **O(N*M*O*I),** kde **N** je počet načítavaných položiek(vek/ vzdelanie), **M** je počet krajov, **O** je počet okresov a **I** je počet obcí.

Kritériá

Názov

Zložitosť je **konštantná** Vráti atribút názov danej ÚJ.

Typ

Pomocou while cyklu aktualizuje nadradenú ÚJ, pokým taká existuje. Zároveň si indexuje, koľko nadradených ÚJ existuje. Potom pomocou switchu a zisteného indexu vypíše typ ÚJ. Zložitosť je **konštantná.**

Príslušnosť

Zložitosť je **konštantná**. Pomocou atribútu nadradenej ÚJ overí, či sa zhoduje so zadaným parametrom.

VzdelaniePočet

Zložitosť je **konštantná**. Pomocou atribútu vzdelanie v ÚJ pristúpi na daný prvok poľa a vráti hodnotu.

VzdelaniePodiel
Zložitosť je **konštantná**. Pomocou atribútov vo vzdelaní v ÚJ zistí počet vzdelaných pre danú úroveň. Celkový počet ľudí v ÚJ (tiež z atribútu vzdelanie), a vypočíta percento.

VekPočet
Zložitosť je **konštantná**. Pomocou atribútu vek v ÚJ pristúpi na daný prvok v poli a vráti hodnotu.

VekPočetOdDo
Zložitosť je **O(N),** kde N je počet vekov, ktoré chceme zistiť. Podľa parametra (muž, ženy alebo oboje) prechádza vo for cykle dané pole a pripočítava ho do premennej.

VekPodielOdDo
Zložitosť je **O(N**), kde N je zložitosť volaného kritéria. Zavolá kritérium **VekPočetOdDo** , ďalej získa počet všetkých obyvateľov z atribútu a vypočíta percento.

VekováSkupinaPočet
Funguje ako kritérium **VekPočetOdDo**, avšak ako parameter dostáva vekovú skupinu. Tá ma presne definovaný parameter od – do. Zložitosť je **O(N**). Kde **N** je počet rokov vo vekovej skupine (t.j. pre produktívny vek (15 - 65) 50).

VekováSkupinaPodiel
Funguje ako kritérium **VekPodielOdDo**, avšak nepracuje s kritérium, ale cez for each si získa vek pre danú skupinu. Potom ho vydelí počtom všetkých obyvateľov v ÚJ. Zložitosť je **O(N).** Kde N je počet rokov vo vekovej skupine (t.j. pre produktívny vek (15 - 65) 50).

Nadradená
Zložitosť je **konštantná**. Vráti atribút ÚJ.

Filtre

Aplikácie obsahuje filtre, ktoré pracujú s jednou hodnotou alebo filtre v rozpätí. Filter s hodnotou je taký, ktorý overuje, či sa kritérium ÚJ rovná práve zadanej hodnote. Filter v rozpätí kontroluje, či sa nachádza hodnota kritéria ÚJ v danom rozpätí. Filter má v konštruktore parameter pre jednu hodnotu/ rozpätie a dané kritérium. Potom je mu možné poslať do metódy belongsTo ÚJ, a vráti sa nám bool hodnota či patrí / nepatrí do danej hodnoty / rozpätia. Zložitosť sa rovná zložitosti kritéria.

Použitie filtra

Pre zložený filter sa vytvorí ArrayList typu Filter. Do neho sa vkladajú filtre. Filtre obsahujú kritériá a všetky hodnoty zadáva používateľ. Pri samotnom filtrovaní sa prechádzajú všetky ÚJ, ktoré je potrebné vyfiltrovať. Každá z týchto ÚJ prechádza vnoreným for each cyklom. Ten prechádza všetky Filtre v ArrayListe. Pomocou metódy belongsTo vo filtri zistíme, či ÚJ spĺňa všetky filtre a môžeme s ňou ďalej pracovať.

Triedenie

Triedime pomocou quick sortu, ten je upravený a v metóde sort naviac prijíma aj kritérium. Tým pádom porovnávaná hodnota, podľa ktorej triedime sa získava pomocou kritéria.

Zložitosť operácií

Bodové vyhľadávanie

Zložitosť vyhľadania ÚJ podľa názvu je **O(log N + k). N** je počet všetkých ÚJ, tie sa nachádzajú v Sorted Sequenced Table, zložitosť vyhľadania v tabuľke je **O(log N). K** je zložitosť vypisovania kritérií. V použitej 4. úrovni sú kritéria pre vypísanie počtov obyvateľov všetkých vekových skupín a počet obyvateľov vo všetkých kategórií nachádzajúcich sa vo vzdelaní.

Filtrovanie podľa typu a prislúchajúcej ÚJ

Pri filtrovaní podľa typu rozlišujeme o aký typ sa jedná, pre obec a okres vrátime danú príslušnú SST tabuľku štátu, čo je **konštantná** zložitosť. Pre kraj, ktorý je inicializovaný ako SST Krajov kvôli jeho podradeným atribútom nemôžeme vrátiť SST, lebo je typu Kraj. Takže v prípade vrátenia kraju pomocou for eachu pridáme prvky do novej SST. Zložitosť je v tomto prípade **O(N).** Kde **N** predstavuje všetky kraje v štáte, ktoré treba prejsť cyklom for each.

V prípade filtrovania ÚJ, ktoré prislúchajú pod užívateľom zadanú ÚJ sa jedná o zložitosť **O(log N+M). O(log N)** predstavuje vyhľadanie zadanej ÚJ v SST všetkých ÚJ v štáte. **M** predstavuje všetky ÚJ, ktoré spadajú pod vyhľadanú ÚJ. Tie treba všetky prejsť cyklom for each.

Filtrovanie s použitím typu ÚJ, ktoré prislúchajú pod užívateľom zadanú ÚJ sa nachádza v sekcií spolu s kódom, ktorý je opísaný v predchádzajúcom odseku. Avšak jedná sa o zložitosť **O(log N + M).** Avšak **M** je tento krát ešte menšie, keďže typ ÚJ, ktorý nechceme pridať sa v cykle for each neprechádza.

Aplikovanie viacerých filtrov

Do zloženého filtra vstupujú dáta už **vyfiltrované podľa typu a prislúchajúcej ÚJ.** Môže nastať, že vyfiltrované nie sú vôbec a používateľ pracuje so všetkými ÚJ. V tomto filtri sa teda pracuje s **N** (ne)vyfiltrovanými ÚJ, tie všetky musíme prejsť v cykle for each. Zložitosť je **O(N * (M + k)). M** predstavuje počet zvolených filtrov ktoré sa musia všetky prejsť v cykle for each. **K** predstavuje zložitosť daného kritéria vo filtri.

Triedenie

Do triedenia tak isto vstupujú dáta **vyfiltrované podľa typu a prislúchajúcej ÚJ.** Tých je **N** a všetky sa musia prejsť v cykle for each. Prv ich vložíme do **Unsorted Sequenced Table**. Vo for each cykle prejdeme **N** údajov vstupných ÚJ. Potom sa podľa voľby vyfiltrujú, avšak v tomto prípade už len maximálne podľa jedného zvoleného filtra, čiže zložitosť je **O(N + k).** Kde **k** je zložitosť daného zvoleného kritéria vo filtri. Potom sa vykoná samotné triedenie, ktoré má zložitosť **O(N* log N).** Keďže sa jedná o quick sort, **N** predstavuje počet vyfiltrovaných dát.

Vyhľadanie najväčšej / najmenšej vekovej skupiny

Do vyhľadania najlepšej / najmenšej vekovej skupiny rovnako vstupujú dáta **vyfiltrované podľa typu a prislúchajúcej ÚJ.** Tých je **N** a všetky sa musia prejsť v cykle for each. Tento cyklus for each sa prechádza a porovnáva či nenašiel najlepšiu / najhoršiu ÚJ. Zložitosť je teda **O(N + k),** kde **k** predstavuje zložitosť zvoleného kritéria.

*Zložitosť výpisu výsledných dát do konzoly je **O(N + k). N** predstavuje počet výsledných dát a **k** v prípade voľby kritéria (keďže nie všade sa vypisuje) predstavuje zložitosť kritéria. V niektorých filtroch sa vypisujú aj nadradené ÚJ a ich typy.

Záver

V semestrálnej práci som si uvedomil dôležitosť použitia štruktúr pre rýchlosť danej problematiky. Zároveň som sa rozhodoval, či zrýchlime aplikáciu, alebo ušetríme pamäťovú náročnosť. Vo veľa systémoch sa musí jedna z týchto vlastností zanedbať v prospech druhej. Aj preto je dôležitá znalosť údajových štruktúr. Je potrebné poznať ako sa bude aplikácia využívať a dobre ju navrhnúť od začiatku do konca. To nám môže radikálne ovplyvniť rýchlosť operácií.