Algoritmy a údajové štruktúry 1

Semestrálna práca

Aplikácia na spracovanie údajov zo Sčítania obyvateľov domov a bytov z roku 2021

Erik Urban

5ZYI24

**Návrh aplikácie z pohľadu použitých údajových štruktúr**

Údajové štruktúry, ktoré som použil na vytvorenie Aplikácie na spracovanie údajov zo Sčítania obyvateľov domov a bytov z roku 2021 sú:

**Utriedená sekvenčná tabuľka(SortedSequenceTable),  
Neutriedená sekvenčná tabuľka(UnsortedSequenceTable),  
Zoznam poli(ArrayList),  
Zreťazený zoznam(LinkedList).**

**Utriedená sekvenčná tabuľka(SortedSequenceTable),**– Utriedené sekvenčné tabuľky som použil na uchovávanie územných jednotiek(obce, okresy, kraje, štát) a ako jedinečné kľúče som použil ich kódy – toto bolo veľmi výhodne pri načítavaní dát a zisťovaní nadradenosti územnej jednotky (časť kódu vždy hovorila o tom, aká je jej nadradená územná jednotka). S touto tabuľkou spolupracovala druhá Utriedená sekvenčná tabuľka, ktorá uchovávala názvy územných jednotiek ako kľúč a dáta boli kódy. Zvolil som utriedenú sekvenčnú tabuľku preto, lebo som potreboval čo najrýchlejšie bodové vyhľadávanie.

**Neutriedená sekvenčná tabuľka(UnsortedSequenceTable),–** Neutriedenú sekvenčnú tabuľku som využil pri vkladaní prvkov pri filtrácií, keďže vkladanie do neutriedenej sekvenčnej tabuľky je rýchlejšie ako vkladať ich do utriedenej sekvenčnej tabuľky. Neutriedenú sekvenčnú tabuľku som taktiež využil pri triedení podľa určitého kritéria.

**Zoznam poli(ArrayList), –** som výhodne využil na ukladanie informácií o vzdelaní a EVS pre danú Územnú jednotku. Veľmi vhodný bol rýchli prístup k dátam podľa indexu.

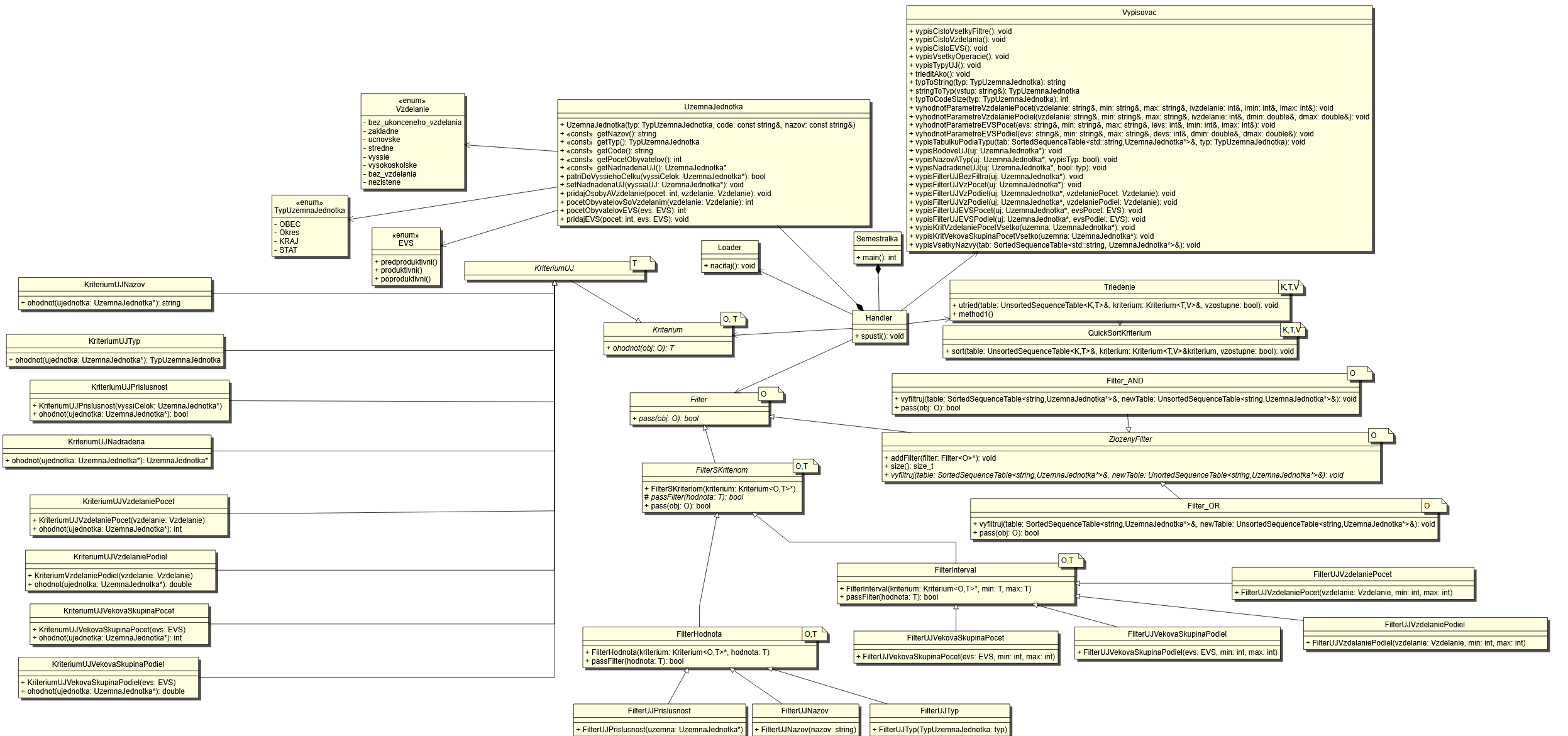
**Zreťazený zoznam(LinkedList) –** táto štruktúra slúžila ako zoznam užívateľom vybraných filtrov. Zvolil som ju preto, lebo som nepotreboval pristupovať k dátam podľa indexu a pohodlnejšej práci s objektami.

**Spracovanie súborov**

Po zvážení kladov a záporov som sa rozhodol pre odstránenie diakritiky vo vstupných dátach. Dôvodom bola mne bližšia práca s typom string(namiesto wstring a jemu prispôsobené metódy). Taktiež mi odpadla povinnosť riešiť preťaženie operátorov porovnania kvôli tomu, že písmena s diakritikou sa nenachádzajú v ASCII tabuľke. Takto je v práci uskutočnené abecedné zoraďovanie podľa anglickej abecedy.

Taktiež som musel zabezpečiť unikátnosť názvov, preto som ku každej duplicite pridal aj jej nadradenú územnú jednotku vo forme napr. Píla – Lučenec , kde Píla je názov obce a Lučenec je názov vyššej územnej jednotky, čiže názov okresu. Tiež bolo potrebné zmeniť názov Zahraničie na Zahraničie1 a Zahraničie2 a okresy Zahraničie na ZahraničieO1 a ZahraničieO2.  
Taktiež ich kódy, ktoré nevyhovovali boli zmenené, ale to až v programe.

**UML diagram**



**Popis tried**

**Main –** v tejto triede sa nachádza metóda main, ktorá je vstupným bodom programu. Nachádza tu inštancia triedy Handler, po vytvorení jej inštancie sa zavolá jej metóda spusti.  
Metóda spusti je v try-catch bloku pre prípadne vyhodenie výnimky (hlavne kvôli tomu, ak by došlo k neotvorení súboru) a bezpečné ukončenie programu.

**Handler –** táto trieda je jadrom programu. Slúži na komunikáciu medzi užívateľom a na komunikáciu medzi jednotlivými časťami programu. Obsahuje metódu spusti. V nej sa vytvorí inštancia triedy Loader a Vypisovač.

**Vypisovac –** tu sa nachádzajú všetky metódy slúžiace na výpis string literálov alebo kriterií.

**Loader –** tu prebieha načítanie dát z csv súborov a transformácia týchto dát na UzemneJednotky. Taktiež sa tu menili kódy pre obec Zahraničie, a okres Zahraničie, ktoré nebolo v správnom formáte. Je tu aj vytvorená územná jednotka Slovensko s kódom "SK".

**Triedenie** – tu prebieha triedenie neutriedenej sekvenčnej tabuľky podľa triediaceho algoritmu, podľa daného kritéria. Na to slúžila inštancia triedy QucikSortKriterium

**QuickSortKriterium –** trieda ktorá použije quick sort algoritmus, triedenie je vykonané podľa daného kritéria.

**Filter** – abstraktná trieda s abstraktnou metódou pass, vracia true alebo false, ak prvok do nej vložený spĺňa podmienky pre filter.

**FilterSKriteriom** – trieda dedí z triedy Filter, má protected metódu passFilter, ktorú majú potomkovia za úlohu prekryť a zistiť, či sa daná hodnota zhoduje s parametrom filtra alebo patrí do intervalu definovaným parametrami filtra. Metóda pass potom vracia bool, ktorý hovorí či objekt vyhovuje danému kritérium. Má na starosti aj deštrukciu všetkých kritérií, ktoré preberájú aj jeho potomkovia ako atribút z parametrického konštruktora.

**FilterHodnota**– Trieda pre filtrovanie podľa určitej hodnoty, táto hodnota je privátny atribút. Dedí FilterSKriteriom.

**FilterInterval** – Trieda pre filtrovanie podľa určitého interválu, obe tieto hodnoty budú ako privátny atribút. Dedí FilterSKriteriom.

**ZlozenyFilter** – Abstraktná trieda s abstraktnou metódou vyfiltruj, ktorú musia prekryť potomkovia.Potom triedy Filter. Má protected atribút, ktorý je LinkedList skladajúci sa s Filtrov. Dovoľuje prijímať filtre a spájať ich do jedného filtra, ktorý využíva všetky prijaté filtre. Na to slúži metóda addFilter. Tiež vie o sebe povedať svoju veľkosť, ktorá sa rovná počtu filtrov v LinkedListe.

**Filter\_AND –** táto trieda prekrýva metódy pass a vyfiltruj tak, aby simulovala matematickú operáciu AND. Metódou pass prejde objekt iba vtedy, ak prejde všetkymi filtrami. Potom sa tieto objekty vložia do novej neutriedenej tabuľky ako objekty ktoré prešli filtráciou. Výnimočný stav je vtedy, ak neobsahuje žiadne filtre, vtedy sa všetky objekty vložia do novej neutriedenej tabuľky.

**Filter\_OR –** táto trieda prekrýva metódy pass a vyfiltruj tak, aby simulovala matematickú operáciu OR. Metódou pass prejde objekt vtedy, ak prejde aspoň jedným filtrom. Potom sa tieto objekty vložia do novej neutriedenej tabuľky ako objekty ktoré prešli filtráciou. Výnimočný stav je vtedy, ak neobsahuje žiadne filtre, vtedy sa všetky objekty vložia do novej neutriedenej tabuľky.

**FilterUJNazov –** potomok triedy FilterHodnota, filtruje podľa názvu vloženého do parametrického konštruktora, spolupracuje s KriteriumNazov.

**FilterUJPrislusnost –** potomok triedy FilterHodnota, filtruje podľa toho či územná jednotka patrí pod príslušnosť územnej jednotky ktorá bola zadaná do parametrického konštruktora, spolupracuje s KriteriumUJPrislusnost.

**FilterUJTyp –** potomok triedy FilterHodnota, filtruje podľa toho či územná jednotka je rovnakého typu ako je typ územnej jednotky, ktorý bol zadaný do parametrického konštruktora, spolupracuje s KriteriumUJTyp.

**FilterUJVekovaSkupinaPocet–** potomok triedy FilterInterval, filtruje podľa toho či dané EVS územnej jednotky je v zadanom intervale, ktorý bol zadaný do parametrického konštruktora, spolupracuje s KriteriumVekovaSkupinaPocet.

**FilterUJVekovaSkupinaPodiel–** potomok triedy FilterInterval, filtruje podľa toho či dané EVS územnej jednotky je v zadanom intervale, ktorý bol zadaný do parametrického konštruktora, spolupracuje s KriteriumVekovaSkupinaPodiel.

**FilterUJVzdelaniePocet–** potomok triedy FilterInterval, filtruje podľa toho či dané Vzdelanie územnej jednotky je v zadanom intervale, ktorý bol zadaný do parametrického konštruktora, spolupracuje s KriteriumUJVzdelaniePocet.

**FilterUJVzdelaniePocet–** potomok triedy FilterInterval, filtruje podľa toho či dané Vzdelanie územnej jednotky je v zadanom intervale, ktorý bol zadaný do parametrického konštruktora, spolupracuje s KriteriumUJVzdelaniePodiel.

**TypUzemnaJednotka** – tento enum definuje možné typy územnej jednotky.

**Vzdelanie** – tento enum definuje možné vzdelania.

**EVS** – tento enum definuje možné EVS.

**UzemnaJednotka** – táto trieda obsahuje všetky dôležité informácie o územnej jednotke. Obsahuje const gettre, ArrayListy vzdelaní a EVS, ktoré v konštruktore inicializuje a zväčší na požadovanú veľkosť. Má parametrický konštruktor, kde príjma typUzemnejJednotky, string kod a string nazov. Sú tu tiež metódy na pripočítavanie ľudí s daným EVS alebo Vzdelaním.

**Popis tried a zároveň popis vyhľadávacích resp. triediacich kritérií :**

**Kriterium –** Kriterium je abstraktná trieda s abstraktnou triedou ohodnoť, ktorú musí implementovať každý potomok.

**KriteriumUJ –** abstraktná trieda, ktorá dedí Kriterium a je predkom všetkých kritérií aplikovateľných na územné jednoty.

**KriteriumUJNazov-** má prekrytú metódu ohodnot, ktorá vráti názov územnej jednotky ako kritérium, daný názov získame getterom getTyp.

**KriteriumUJTyp -** má prekrytú metódu ohodnot, ktorá vráti typ územnej jednotky ako kritérium, daný názov získame getterom getTyp.

**KriteriumUJPrislusnost –** má prekrytú metódu ohodnot, ktorá vracia bool hodnotu, podľa toho, či územná jednotka patrí do vyššieho územného celku, daný vyšší územný celok je inicializovaný pri vytvorení inštancie, má parametrický konštruktor, kde parametrom je smerník na vyšší celok. Príslušnosť zistíme cez metódu patriDoVyssiehoCelku, ktorá ma parameter, ktorý preberá smerník na UzemnuJednotku.

**KriteriumUJNadradena** – má prekrytú metódu ohodnot, ktorá vracia smerník na priamo nadradenú územnú jednotku. Získame ju cez getter getNadriadenaUJ.

**KriteriumUJVzdelaniePocet –** má prekrytú metódu ohodnot, ktorá vráti celočíselnú hodnotu, ktorá určuje počet ľudí s daným vzdelaním. Dané vzdelanie je prebraté cez parametrický konštruktor, následne uložené ako privátny atribút. Hodnotu získame pomocou metódy pocetObyvatelovSoVzdelanim kde ako parameter je vložené dané vzdelanie.

**KriteriumUJVzdelaniePodiel -** má prekrytú metódu ohodnot, ktorá vráti percentuálny podiel ľudí s daným vzdelaním, zadaným ako kritérium vyhľadávania. Dané vzdelanie je prebraté cez parametrický konštruktor, následne uložené ako privátny atribút. Hodnotu získame pomocou metódy pocetObyvatelovSoVzdelanim s parametrom vzdelania, jej vynásobením literálom 100 a následne podelením počtom obyvateľov. Počet obyvateľov zistíme pomocou getteru getPocetObyvatelov.

**KriteriumUJVekovaSkupinaPocet –** má prekrytú metódu ohodnot, ktorá vráti celočíselnú hodnotu, ktorá určuje počet ľudí s daným EVS. Dané EVS je prebraté cez parametrický konštruktor, následne uložené ako privátny atribút. Hodnotu získame pomocou metódy pocetObyvatelovEVS kde ako parameter je vložené dané EVS.

**KriteriumUJVekovaSkupinaPodiel –** má prekrytú metódu ohodnot, ktorá vráti percentuálny podiel ľudí s daným EVS, zadaným ako kritérium vyhľadávania. Dané EVS je prebraté cez parametrický konštruktor, následne uložené ako privátny atribút. Hodnotu získame pomocou metódy pocetObyvatelovEVS s parametrom EVS, jej vynásobením literálom 100 a následne podelením počtom obyvateľov. Počet obyvateľov zistíme pomocou getteru getPocetObyvatelov.

**Zložitosti všetkých operácií**

**Zložitosť načítania dát:**

Otvorenie súboru – O(1), Zatvorenie súboru – O(1), string operácia assign O(1), vytvorenie novej územnej jednotky – O(1), Vloženie prvku do utriedenej sekvenčnej tabuľky table- O(1) -lebo dáta sú už zoradené podľa kľúča,Vloženie prvku do utriedenej sekvenčnej tabuľky kodovanie- O(N) , čítanie zo súboru k – počet stĺpcov a N počet riadkov, čiže O(k\*N) = O(N), metóda stoi – O(1), pristúpenie na index ArrayListu – O(1),funkcia substr o konštantný počet charakterov – O(1), Funkcie na pridávanie a settovanie – O(1), nájdenie prvku v sekvenčnej tabuľke O(log(N)).

Spracovanie dát Kraje, Okresy, Obce, Vzdelaní a Vekov odhadujem hornú asymptotickú zložitosť na **O(k \* N \* M \* log(P))**, kde k je konštantný počet stĺpcov súbora, N je počet riadkov, M je počet prvkov v tabuľke kódovanie a P je počet prvkov v tabuľke table.

**Zložitosť bodové vyhľadávanie:**

Zložitosť bodového vyhľadávania v mojej aplikácii je O(2\*log(N)), kde N je počet prvkov v tabuľke,keďže najprv vyhľadávam v tabuľke, kde je názov kľúč a kód sú dáta a potom vyhľadávam v tabuľke všetkých územných jednotiek, kde je daný kód kľúčom. Po zanedbaní konštánt výsledná horná asymptotická zložitosť je **O(log(N)),** kde n je počet prvkov v tabuľke.

**Zložitosť filtrovanie:**

Zložitosť filtrovania je O(k \* N \* m), kde k je konštantný počet filtrov v LinkedListe a N je počet prvkov v tabuľke a m je počet prvkov ktoré sa pridajú do neusporiadanej sekvenčnej tabuľky, kde pridávanie je O(1)).   
Po zanedbaní konštánt je horná asymptotická zložitosť filtrovania **O(N)**.

**Zložitosť triedenie:**

Na vykonanie triedenia tabuľky podľa určitého kritéria bol použitý QuickSort algoritmus, ktorý mi zabezpečí **O(N \* log2(N))** hornú asymptotickú zložitosť, kde N je počet prvkov v tabuľke.

**Používateľská príručka**

**Po spustení programu** užívateľ krátku dobu čaká na načítanie dát o čom je aj informovaný.  
Následne sa na konzole objaví zoznam operácií, z ktorých si užívateľ môže vyberať.  
Program vyzýva užívateľa väčšinou na číselný input pre voľbu žiadanej operácie. Každá operácia musí byť potvrdená ENTER-om!

**Bodové vyhľadávanie** si najprv vyžiada typ územnej jednotky a následne žiada názov určitej územnej jednotky v presnom tvare, ktorým je oficiálny názov územnej jednotky. Názov musí byť platný pre daný typ územnej jednotky. Okresy majú slovo 'Okres' pred svojim oficiálnym názvom. Kraje majú zas slovo kraj za svojim oficiálnym názvom. Takže je rozdiel napr. medzi 'Žilina' a 'Okres Žilina' a 'Žilinský kraj'. Prvé určuje obec, ďalšie okres a posledné kraj. Duplicity majú navyše pridané k svojmu názvu aj priamu nadradenú jednotku. Takže ak máme napr. obec Píla, ktorá má duplicitu tak musí užívateľ napísať bez apostrofov a medzerami  
'Píla – Lučenec' , kde Píla je názov obce a Lučenec je názov vyššej územnej jednotky, čiže názov okresu.

**Pri filtrácii** si užívateľ vyberá najprv či chce alebo nechce použiť filtre. Ak nechce, tak sa vypíšu všetky územne jednotky zoradené podľa ich kódu. Vypíšu sa ich názvy, typ a názvy a typy nadriadených jednotiek danej územnej jednotky. Ak chce užívateľ používať filtre, je vyzvaný k tomu, aby si vybral, či chce AND alebo OR logickú operáciu. Pri AND to znamená, že územná jednotka musí spĺňať všetky filtre, aby bola vložená do vyfiltrovanej tabuľky. Pri OR stačí, ak spĺňa aspoň jeden filter. Po vybratí logickej operácie si môže užívateľ vybrať zo zoznamu filtrov, ktoré budú aplikované.   
Potom už stačí stlačiť a potvrdiť tlačidlo 7, pre 'Uz nevyberat'. Následne sa vypíšu všetky územné jednotky, ktoré vyhovujú filtrom, podľa vybranej logickej operácie.

**Pri triedení** je užívateľ vyzvaný vybrať si najprv zo spoločných filtrov a potom ďalej z filtrov, s kritériom podľa ktorých sa bude triediť. Po vyfiltrovaní je ešte užívateľ vyzvaný k tomu, či chce utriediť tabuľku vzostupne alebo zostupne. Potom sú už územné jednotky vypísané na konzolu podľa vybraných parametrov.

**Posledná operácia je ukončenie aplikácie**, ktorú keď si zvoli užívateľ ukončí aplikáciu.