

SEMESTRálna práca

Algoritmy a údajové štruktúry 1 Tomáš Kotrík 559142



29. mája 2022

Žilinská univerzita

Fakulta Riadenia a Informatiky

Obsah

[Zadanie Semestrálnej práce 2](#_Toc104743183)

[UML DIAGRAM 4](#_Toc104743184)

[Popis Tried 5](#_Toc104743185)

[Spracovanie pôvodných dát 10](#_Toc104743186)

[Použitie štruktúr a analyzovanie zadania 11](#_Toc104743187)

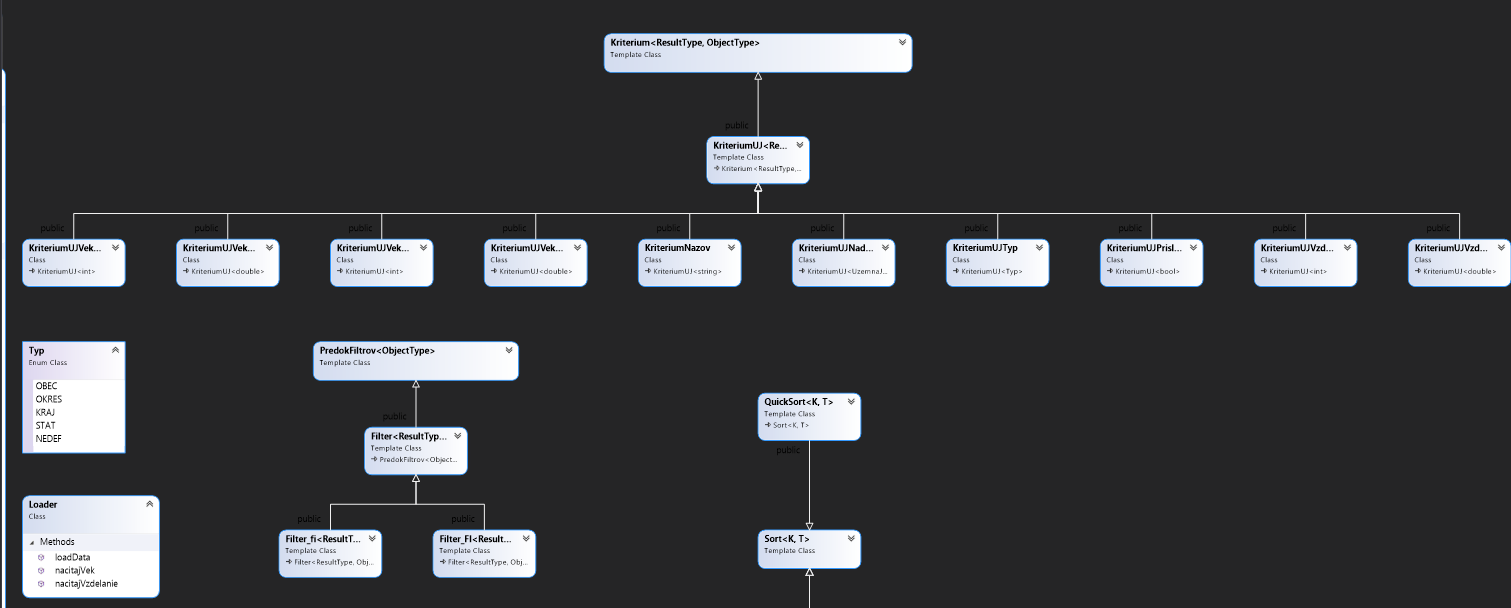
[Zložitosti Operácii 14](#_Toc104743188)

[Prehľad o vypracovanosti zadania 17](#_Toc104743189)

[Používateľská príručka 18](#_Toc104743190)

# Zadanie Semestrálnej práce

Vytvorte aplikáciu, ktorá umožní spracovať výsledky zo sčítania obyvateľov domov a bytov, ktoré sa konalo v roku 2021.

UML DIAGRAMObrázok, na ktorom je text

Automaticky generovaný popis

# Popis Tried

1. UzemnaJednotka.h

atribúty:

class UzemnaJednotka {

private:

int id\_UJ; // id uzemnej jednotky

int pocet\_Obyvatelov; // pocet obyvatelov UJ

std::string nazov; //nazov UJ

Typ typ\_UJ; // Typ z enumu ciye stat kraj okres obec

UzemnaJednotka\* moje\_Zaclenenie; // zaclenujem sa k predkovi cize ak obec tak moje zaclenenie je v okrese

std::string kodJednotky; //kod jednotky z csv

structures::Array<int>\* vek; //hodnoty z csv vek

structures::Array<int>\* vzdelanie; //hodnoty z csv vzdelanie

structures::Array<int>\* evs; // ekonomicka vekova skupina

metódy:

...gettery a settery daných atribútov

ostatné:

metódy zvysEVS , zvysPocetObyvatelov, zvysVzdelanie, zvysVek zobere z parametra array EVS kedze som neriesil cez enum tak je to odindexovane a do svojho atribútu evs postupne po indexoch naplusuje hodnoty z parametra ...takto postupujem od obce az ku slovensku

inline void zvysEVS(structures::Array<int>\* arrEVS) {}

inline void zvysPocetObyvatelov(structures::Array<int>\* arrEVS) {}

inline void zvysVzdelanie(structures::Array<int>\* arrVZD) {}

inline void zvysVek(structures::Array<int>\* arrVek) {}

metódy atIndex dostanú z parametra index a vráti sa hodnota na danom indexe arrayu

int getEvsAtIndex(int index) {}

int getVzdelanieAtIndex(int index) {}

int getVekAtIndex(int index) {}

slovensko nemá predka takže bude ukazovať na samého seba

UzemnaJednotka\* mojUkazovatel() {}

1. Loader.h

metódy:

loadData je metóda ktorá dostane do parametrov prázdne tabulky obci, okresov a krajov

a postupne ich s csv načíta a vloží do tabuliek.

void loadData(SortedSequenceTable<std::string, UzemnaJednotka\*>\* obce, SortedSequenceTable<std::string, UzemnaJednotka\*>\* okresy,

SortedSequenceTable<std::string, UzemnaJednotka\*>\* kraje);

nacitajVek a nacitajVzdelanie nacita vek a vzdelanie z csv súborov a vloží ich do tabuľky ako kĺúč má názov a hodnoty ma array vekov a vzdelani

SortedSequenceTable<std::string, Array<int>\*>\* nacitajVek();

SortedSequenceTable<std::string, Array<int>\*>\* nacitajVzdelanie();

1. Enums.h

//definoval som si len Typ

//Riesenie muzov zien EVS Vzdelania som riesil cez indexy

enum class Typ {

OBEC,

OKRES,

KRAJ,

STAT

};

1. Kriterium.h

//podla seminaru a prirucky

Predok všettkých kriteríi

class Kriterium {};

class KriteriumUJ : public Kriterium<ResultType, UzemnaJednotka\*> {};

KriteriumNazov vráti pomocou string názov územnej jednotky

class KriteriumNazov : public KriteriumUJ<std::string> {};

KriteriumUJTyp vráti daný typ z enumu územnej jednotky

class KriteriumUJTyp : public KriteriumUJ<Typ> {};

KriteriumUJPrislusnost ... ak kontrolovaná jednotka má zadanú príslušnosť vráti true inak false

class KriteriumUJPrislusnost : public KriteriumUJ<bool> {};

KriteriumUJVzdelaniePocet vráti počet ľudí vzdelaných v danej skupine (indexe)

class KriteriumUJVzdelaniePocet : public KriteriumUJ<int> {};

KriteriumUJVzdelaniePodiel vráti podiel ľudí vzdelaných v danej skupine (indexe)

class KriteriumUJVzdelaniePodiel : public KriteriumUJ<double> {};

KriteriumUJVekPocet vráti počet ľudí zo zadanej hranice napr 0-18 rokov a či muž alebo žena (indexy)

class KriteriumUJVekPocet : public KriteriumUJ<int> {};

KriteriumUJVekPodiel vráti podiel ľudí zo zadanej hranice napr 0-18 rokov a či muž alebo žena (indexy)

class KriteriumUJVekPodiel : public KriteriumUJ<double> {};

KriteriumUJVekovaSkupinaPocet vráti počet ľudí v danej vekovej skupine (index)

class KriteriumUJVekovaSkupinaPocet : public KriteriumUJ<int> {};

KriteriumUJVekovaSkupinaPodiel vráti podiel ľudí v danej vekovej skupine (index)

class KriteriumUJVekovaSkupinaPodiel : public KriteriumUJ<double> {};

KriteriumUJNadradena vráti typ a názov nadradenej územnej jednotky

class KriteriumUJNadradena : public KriteriumUJ<UzemnaJednotka\*> {};

1. Filter.h

class PredokFiltrov {

//vlastny predok aby som vedel nasledne lepsie pracovat v liste pri vyhladavani

class Filter : public PredokFiltrov<ObjectType> {};

Filter\_fi je filter s hodnotou pre kritéria názvu, typu a príslušnosti

class Filter\_fi : public Filter<ResultType, ObjectType> {};

Filer\_FI je intervalový filter pre všetky ostatné kritéria

class Filter\_FI : public Filter<ResultType, ObjectType> {};

1. Vyhladavanie.h

atribúty:

int volba; voľba pre zvolené číslo vstupu užívateľa

int sortVolba = 0; sortVolba pre zvolený vstup pri triedení

string strasnePeknyRamik = "\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*"; orámovanie

int iterator = 0; iterátor pre iterovanie medzi krajmi vo for

int iteratorOkresov = 0; iterátor okresov pre cyklus

int iteratorObcii = 0; iterátor obcí pre cyklus

int counterPreSkipnutieSlovenska = 0; slovensko mam v tabuľke krajov keď ho nechcem vypisovať v krajoch musím ho skipnúť

int vybrateKriteriumPreFilter = 0; zadané kritérium na mapovanie

int alpha = 0; filter alpha

int beta = 0; filter beta

int indexVzdelaniaPocet = 0; index vzdelania pocet pre triedenie

int indexVzdelaniaPodiel = 0; index vzdelania podiel pre triedenie

int counterPreVypisovanieFiltrov = 0; napočítavanie pre výpis filtrov

string vybranyKraj = ""; výber podľa užívateľa

string vybranyOkres = ""; výber podľa užívateľa

string vybranaObec = ""; výber podľa užívateľa

string vyber = ""; string vstup od užívateľa keď treba zadať slovne

string cakamKlavesu = "CAKAM KLAVESU >>> ";

int cislo = 0; číslo na mapovanie volieb a výpis

bool opakuj = true; pre cyklus orientovanie

//arrayList filtrov pre prechadzanie a kontrolu bool splnaFilter

ArrayList <PredokFiltrov<UzemnaJednotka\*>\*>\* listFiltrov = new ArrayList<PredokFiltrov<UzemnaJednotka\*>\*>();

//vystupna tabulka ktora sa bude sortovat

UnsortedSequenceTable<std::string, UzemnaJednotka\*>\* vystupnaTabulka;

int pocetFiltrov = 0;

//KRITERIA

KriteriumNazov\* kriteriumNazov = new KriteriumNazov();

KriteriumUJVzdelaniePocet\* kriteriumUJVZPocet = new KriteriumUJVzdelaniePocet(NULL);

KriteriumUJVzdelaniePodiel\* kriteriumUJVZPodiel = new KriteriumUJVzdelaniePodiel(NULL);

KriteriumUJTyp\* kriteriumUJTyp = new KriteriumUJTyp();

KriteriumUJNadradena\* kriteriumUJNadradena = new KriteriumUJNadradena();

KriteriumUJPrislusnost\* kriteriumUJPrislusnost = new KriteriumUJPrislusnost(nullptr);

KriteriumUJVekPocet\* kriteriumUJVekPocet = new KriteriumUJVekPocet(0);

KriteriumUJVekPodiel\* kriteriumUJVekPodiel = new KriteriumUJVekPodiel(0);

KriteriumUJVekovaSkupinaPocet\* kriteriumUJVekovaSkupinaPocet = new KriteriumUJVekovaSkupinaPocet(NULL);

KriteriumUJVekovaSkupinaPodiel\* kriteriumUJVekovaSkupinaPodiel = new KriteriumUJVekovaSkupinaPodiel(NULL);

//FILTRE fi

Filter\_fi<std::string, UzemnaJednotka\*>\* filterNazov = new Filter\_fi<std::string, UzemnaJednotka\*>(kriteriumNazov);

Filter\_fi<Typ, UzemnaJednotka\*>\* filterUJTyp = new Filter\_fi<Typ, UzemnaJednotka\*>(kriteriumUJTyp);

Filter\_fi<bool, UzemnaJednotka\*>\* filterUJPrislusnost = new Filter\_fi<bool, UzemnaJednotka\*>(kriteriumUJPrislusnost);

//FILTRE FI

Filter\_FI<int, UzemnaJednotka\*>\* filterVzdelaniePocet = new Filter\_FI<int, UzemnaJednotka\*>(kriteriumUJVZPocet);

Filter\_FI<double, UzemnaJednotka\*>\* filterVzdelaniePodiel = new Filter\_FI<double, UzemnaJednotka\*>(kriteriumUJVZPodiel);

Filter\_FI<int, UzemnaJednotka\*>\* filterVekPocet = new Filter\_FI<int, UzemnaJednotka\*>(kriteriumUJVekPocet);

Filter\_FI<double, UzemnaJednotka\*>\* filterVekPodiel = new Filter\_FI<double, UzemnaJednotka\*>(kriteriumUJVekPodiel);

Filter\_FI<int, UzemnaJednotka\*>\* filterVekovaSkupinaPocet = new Filter\_FI<int, UzemnaJednotka\*>(kriteriumUJVekovaSkupinaPocet);

Filter\_FI<double, UzemnaJednotka\*>\* filterVekovaSkupinaPodiel = new Filter\_FI<double, UzemnaJednotka\*>(kriteriumUJVekovaSkupinaPodiel);

metódy:

konštruktor pre vyhladavanie

Vyhladavanie(SortedSequenceTable<std::string, UzemnaJednotka\*>\* krajeParam,

SortedSequenceTable<std::string, UzemnaJednotka\*>\* okresyParam,

SortedSequenceTable<std::string, UzemnaJednotka\*>\* obceParam);

deštruktor

~Vyhladavanie();

vypis kritérii pre bodové vyhľadávanie

void hodnotyKriterii(TableItem<std::string, UzemnaJednotka\*>\* item);

vypíše kritérium názvu

void vypisKriteriaNazov(UzemnaJednotka\* uj);

vypíše kritérium Typu

void vypisKriteriaUJTyp(UzemnaJednotka\* uj);

vypíše kritérium Príslušnosti

void vypisKriteriaUJPrislusnost(UzemnaJednotka\* uj);

vypíše kritérium Nadradenosti

void vypisKriteriaNadradena(UzemnaJednotka\* uj);

vypíše kritérium počtu vypíše kritérium názvu Vzdelanosti

void vypisKriteriaUJVzdelaniePocet(UzemnaJednotka\* uj, int indexVzdelania);

vypíše kritérium podielu Vzdelanosti

void vypisKriteriaUJVzdelaniePodiel(UzemnaJednotka\* uj, int indexVzdelania);

vypíše kritérium počtu Veku

void vypisKriteriaUJVekPocet(UzemnaJednotka\* uj, int indexVzdelania);

vypíše kritérium podielu Veku

void vypisKriteriaUJVekPodiel(UzemnaJednotka\* uj, int indexVzdelania);

vypíše kritérium počtu danej Vekovej Skupiny

void vypisKriteriaUJVekovaSkupinaPocet(UzemnaJednotka\* uj, int evsIndex);

vypíše kritérium podielu danej Vekovej Skupiny

void vypisKriteriaUJVekovaSkupinaPodiel(UzemnaJednotka\* uj, int evsIndex);

vypíše spracované filtre

void vypisFilter(std::string zvolenaUJ, int mozem, SortedSequenceTable<std::string, UzemnaJednotka\*>\* zvolenaTabulka, SortedSequenceTable<std::string, UzemnaJednotka\*>\* prislusnost);

hranice alfa beta pre filtrovanie v prípade hodnostných filtrov tak nazov a uj

void zaznamenajHraniceFiltrov(int volba, std::string nazov, UzemnaJednotka\* uj, int spodnaHranica, int hornaHranica);

pridá podľa užívateľa dané filtre do listu

void pridajDoListuFiltrov(int volbaPodla);

vypisovanie pre triedenie konečného výstupu

void sortOtazky();

utriedi výstupnu tabuľku quickSortom

void sortujNeutriedenuTabulku(UnsortedSequenceTable<std::string, UzemnaJednotka\*>\* neutriedena, int podlaCoho, bool vzostupne);

1. Source.cpp

initHeapMonitor() pre kontrolovanie memory leakov.

Následné vytvorenie práznych tabuliek pre obce, kraje, okresy, vek a vzdelanie.

Oznámenie užívateľovi o vyčkaní a udanie stavu v ktorom sa nachádza

Napokon od obcí ku svojim predkom až ku slovensku navýšenie údajov o obyvateľoch.

Obrázok, na ktorom je text

Automaticky generovaný popis

Spustenie triedy vyhľadávanie kde sa rieši celý konzolový interface.

A následné deštruktory aby ostala pamäť čistá.

Obrázok, na ktorom je text

Automaticky generovaný popis

# Spracovanie pôvodných dát

Údaje, s ktorými má aplikácia pracovať je možné nájsť v archíve CSV.zip uloženom v MS Teams v adresári Semestrálna práca v záložke Súbory v kanáli Semestrálne práce.

Zo všetkých dát som si odstránil diakritiku pomocou nástroju na stránke kde som vložil celý obsah súboru a prekopíroval naspäť do csv súboru

<https://www.alejtech.sk/sk/sluzby/doplnkove-sluzby/online-nastroje-na-upravu-textu/odstranit-diakritiku.html>

Následne som si odstránil zahraničie ktoré vpodstate nebolo potrebné riešiť a nemalo ani žiadne hodnoty.

Duplicity som riešil vnútorne v kóde kde som v prípade duplicít k názvu doplnil aj jeho kód.

# Použitie štruktúr a analyzovanie zadania

Analyzovanie:

* Je zapotrebné spracovať údaje z csv súborov, vložiť ich do vhodných štruktúr a navrhnúť správny algoritmický postup.
* Po pridaní základných údajov je ďalším krokom namapovanie svojích predkov aby vedeli o sebe to znamená pridávanie

Riešenie 1: Slovensko🡨Kraj🡨Okres🡨Obec

Riešenie 2: Slovensko🡪Kraj🡪Okres🡪Obec

Riešenie 3: Slovensko🡨🡪Kraj🡨🡪Okres🡨🡪Obec

* Po vyriešení mapovania predkov a potomkov je potrebné načítať vzdelanie a vek z csv súborov a vložiť ich do vhodných štruktúr
* Ďalším krokom bude priradenie a navýšenie týchto údajov pre predkov a potomkov
* Následne spracovanie dát v konzolovom interface a požadované funkcionality pre kritéria a filtrovanie podľa úrovní

Použitie štruktúr:

**Array** – použil som pre spracovanie EVS, veku a vzdelania.

**ArrayList** – použil som pri filtroch v triede Vyhladavanie.h kde som potreboval riešiť zadané filtre od užívateľa a následné pridanie do listu ktoré si zvolil.

**Utriedená sekvenčná tabuľka** – použil som na údaje z načítaných csv súborov pre obce, okresy, kraje, vzdelanie a vek. Využitie plynulo aj zo zadania kde potrebná horná asymptotická zložitosť pre nájdi prvok v tabuľke mala byť menšia ako O(N). SST toto spĺňa zložitosťou O(log N).

**Neutriedená sekvenčná tabuľka** – na spracovanie výslednej tabuľky z filtrovania kde prebehne utriedenie quick sortom podľa zadaného kritéria.

Spracovanie mojej analýzy:

* Csv súbory som spracovával cez ifstream a pomocou getline načítaval údaje delimiter bola klasicky ‘;’. Na začiatku som si ešte vytvoril Územnú jednotku Slovensko s kódom 100000, názvom Slovensko a Typ STAT ktorej som setol ukazateľ na seba. Slovensko som teda pridal do tabuľky krajov keďže nebolo potrebné mať vlastnú tabuľku na slovensko. Následne prvý riadok nám nebol potrebný takže ten sa celý načítal do prázdneho a ďalej nespracovával. Spracovali sa len hodnoty kód, oficiálny názov a poznámka.
* Môj strategický postup bol teda **Slovensko🡨Kraj🡨Okres🡨Obec**. Z načítaných údajov sme mohli vytvoriť územnu jednotku KRAJ ktorého ID bolo číslo kódu, názov bol oficiálny názov z csv a Typ enumu bol KRAJ. V cykle som sa pýtal tabuľky krajov či už náhodou načítaný kraj nebol pridaný v tabuľke ak nie tak sa insertol. Tak isto sa setol stringový kód a začlenenie ktoré patrilo Slovensku.

Týmto spôsobom som riešil aj okresy a obce. Pri okresoch som si parsol substring kódu podľa čoho som vedel priradiť okres ku kraju a následne priradil. Vytvorená územná jednotka okresu sa teda skladala z kódu ktorý bol substring a parsované 3j ciferné číslo aby som vedel namapovať obce, z názvu daného okresu a napokon Typu OKRES.

Pre obce som kontroloval kód okresu a kód obce či ich kódy sú v prvých 6tich znakov rovnaké ak áno tak som vedel že je to správne priradenie. Vytvorená územná jednotka sa skladala z ID 3jciferného čísla, názvu a Typu OBEC.

* Pre načítanie veku a vzdelania som použil SST kľúčom názov a data boli v Array intov najprv. Prvý riadok zobere na prázdno. Ostatné riadky zobere najprv kód obce a zvyšné hodnoty nahadzuje do arrayu podľa indexu ktorý je od 0 po 202 keďže som potom samostatne pristupoval podľa indexu (od 0 – 101 muzi, od 101 -202 zeny) teda nerozlišoval som Enumom na MUZI a ZENY. Tak isto som to spravil pre Vzdelanie kde bol array veľkosti 8 a indexom som postupoval poďla :
* Navyšovanie pre predkov som riešil teda od obcí kde som postupne vo výpočtových metóda zvýš... zvýšil údaje pre predkov.

Obrázok, na ktorom je text

Automaticky generovaný popis

* Posledné čo je za potrebné je sa vysporiadať s konzolovým interface-om ktorý som celý riešil v triede Vyhladavanie.h

# Zložitosti Operácii

**Načítanie Slovenska do tabuľky krajov:**

*O(N)*

**Načítavanie dát pre Kraje:**

Načítanie do Utriedenej sekvenčnej tabuľky kde zložitosť vkladania je O(N) a kontrola kľúča je cez metódu containsKey() ktorá je O (log N), definujme premennú *k* ako počet údajov v jednom riadku a premennú *q* ako počet riadkov v csv. Takže zložitosť = *k\*q\*O(N) + O(log N)*

**Načítavanie dát pre Okresy:**

Podobne ako pri Krajoch je vkladanie O(N), kontroluje sa správnosť kľúča ale kedže sa znovú hľadá v krajoch jeho názov cez find je to 2 \* O (log N), následne premnné *k* ako počet údajov v jednom riadku a premenná *q* ako počet riadkov v csv. Výsledná zložitosť = *k\*q\*O(N)+2\*O(log N)*

**Načítavanie dát pre Obce:**

Pri Obciach prezerám najprv či už sa zadaný kľúč nenachádza v tabuľke ak áno tak ešte pred insertom ho musím zmeniť aby sa vložil to je prehľadanie tabuľky O(log N), následne kontrolujem unikátnosť kľúča takže ďalšie O (log N), ďalší for each cez tabuľku takže O(log N), napokon treba setnúť začlenenie to znamená ďalší find čiže O (log N) a opäť vkladanie do tabuľky je O(N), premenná *k* ako počet údajov v jednom riadku a premenná *q* ako počet riadkov v csv. Výsledná zložitosť = *k\*q\*O(N)+4\*O(log N)*

**Načítanie dát do tabuľky veku a vzdelania:**

Vkladanie do tabuľky je O(N) prechádzanie zoznamom je O(1) *k* premenná počet riadkov *q* premenná počet údajov v riadku.

Výsledná zložitosť = *k\*q\*O(N)+O(1)*

**Nastavenie a navýšenie hodnôt pre Slovensko🡨Kraj🡨Okres🡨Obec**:

Prehľadávanie tabuľkov obcí je O(log N)

setVek() má O(N) lebo removuje z parametra a pridáva do seba

setVzdelanie je O(N) lebo removuje z parametra a pridáva do seba

setEvs je 2\*O(1) kvôli tomu že dva krát prechádza array cez indexy

zvysVek je 2\*O(1) lebo v jednom cykle načíta prvok z at a priradí do seba at

zvysPocetObyvatelov je O(1) lebo len do seba načíta hodnoty z array pomocou at

zvysEVS je 2\*O(1) tak isto ako vek

zvysVzdelanie je 2\*O(1) tak isto ako vek

A keďže idem od obcí až ku slovensku sa to bude postupne násobiť preto definujme premennú *k* ako koľko krát musí préjst daná obec a nastaviť až po Slovensko.

Výsledná zložitosť by teda mohla vyzerať🡪*k\*O(log N) +2\*O(N)+9\*O(1)*

**Kritéria:**

Kriteriúm Názov – vráti len getter danej jednotky atribútu nazov – *O(1)*

Kritérium UJ Typ – vráti getter atribútu enum typ – *O(1)*

Kritérium UJ Príslušnosť – vráti true alebo false ak sa nájde príslušnosť – *O(1)*

Kritérium Vzdelanie Počet – z gettera prejde array a vráti hodnotu – *O(1)*

Kritérium Vzdelanie Podiel – najprv vráti hodnotu z indexu takže O(1) potom vráti počet obyvateľov – O(1) a vypočíta – *2\*O(1)*

Kritérium Vek Počet – prejde array cez indexy na počíta a vráti – *O(1)*

Kritérium Vek Podiel – tak isto ako pri vzdelaní podiel – *2\*O(1)*

Kritérium EVS Počet – z indexu array vráti hodnotu – *O(1)*

Kritérium EVS Podiel – tak isto ako pri ostatných podieloch – *2\*O(1)*

Kritérium Nadradená – vráti názov nadradenej jednotky *O (1)*

**Filtre:**

Filter fi alebo Filter FI – vždy ide do výstupu nainsertovaná vyfiltrovaná tabuľka takže – *O(N)*

**Triedenie:**

QuickSort triedenie neutriedenej tabuľky zložitosť – *O(N\*log2N)*

**Bodové vyhľadávanie:**

Prehľadávanie tabuliek a vypisovanie 🡪 O(log N) +k\*O(1) kde *k* je premenná počet výpisov

**Výsledok triedenia:**

Ak sa mi vyfiltrovala tabuľka a utriedim ju quick sortom výsledná zložitosť je *O(N)\*O(N\*log2N)*

# Prehľad o vypracovanosti zadania

1. Úroveň 🡪 Bodové vyhľadávanie : Kritérium Názov, kritérium UJ Vzdelanie počet Filtrovanie : Filter vzdelanie počtu a podielu
2. Úroveň 🡪 Triedenie obcí : zoradenie podľa kritéria názov a vzdelanie počet, vzdelanie podiel
3. Úroveň 🡪 Bodové vyhľadávanie: Kritérium názov Kritérium Typ a aj pre vyššie jednotky do ktorých patrí, Kritérium Vzdelanie počet

Filtrovanie: Filter Vzdelanie počet / podiel Filter Typ a Filter Príslušnosť

Triedenie: zoradenie podľa kritérium názov, kritérium vzdelanie počet a vzdelanie podiel

1. Úroveň 🡪 Bodové vyhľadávanie: Kritérium názov, Kritérium UJTyp aj pre nadradené, Kritérium Vzdelanie Počet/Podiel

Filtrovanie: Filter Nazov, Filter Typ, Filter Vzdelanie počet/podiel, Filter Typ, Filter Príslušnosť, Filter Vek počet/podiel, Filter EVS počet/podiel

Triedenie: zoradenie podľa kritéria Vek podiel/vek počet, EVS počet/podiel ....Výber najlepších

# Používateľská príručka

**ŠTART:**

Oboznámenie užívateľa o vyčkaní kým sa načítajú a priradia údaje.

Obrázok, na ktorom je text

Automaticky generovaný popis

Obrázok, na ktorom je text

Automaticky generovaný popis

**Úvodná obrazovka:**

Užívateľ si môže vybrať z 3 možností**Obrázok, na ktorom je text

Automaticky generovaný popis**

**Bodové vyhľadávanie:**

Obrázok, na ktorom je text

Automaticky generovaný popisVypíše sa ako prvé Slovensko ... jeho názov, typ, počet obyvateľov nadradenosť teda ukazuje na seba ako som spomínal a následne prehľad vzdelanostných skupín podľa počtu a podľa podielu **Obrázok, na ktorom je text

Automaticky generovaný popis**

Zobrazia sa kraje kde si môže ďalej vybrať alebo sa vrátiť do menu

Vybrali sme si Banskobystrický kraj a znovu nam ukáže prehľad o ňom

Obrázok, na ktorom je text

Automaticky generovaný popisObrázok, na ktorom je text

Automaticky generovaný popis

Pre daný kraj sú tieto možnosti okresov

Prehľad o okrese

Obrázok, na ktorom je text

Automaticky generovaný popis

Obce pre daný okres

Obrázok, na ktorom je text

Automaticky generovaný popis

A výpis pre danú obec

Obrázok, na ktorom je text

Automaticky generovaný popis

Ak sa chce užívateľ vrátit tak na konci má možnosť



**Filtrovanie:**

Ponúkne používateľovi množiny z ktorých chce filtrovať alebo sa môže vrátiť späť

**Obrázok, na ktorom je text

Automaticky generovaný popis**

Používateľ si môže zadať koľko filtrov chce aplikovať v prípade 0 sa prejde na triedenie celej množiny

**Obrázok, na ktorom je text

Automaticky generovaný popis**

Umožníme užívateľovi si vybrať z následovných filtrov

**Obrázok, na ktorom je text

Automaticky generovaný popis**

Ďalšie možnosti pri vzdelaní

**Obrázok, na ktorom je text

Automaticky generovaný popis**

Vyfiltrovalo nám množinu ďalej sa pýtame či užívateľ chce triediť danú množinu

Obrázok, na ktorom je text

Automaticky generovaný popis

Filtrovať môže podľa Názvu/počtu vzdelania/podielu vzdelania

Vzostupne/Zostupne

**Obrázok, na ktorom je text

Automaticky generovaný popis**

**Ukončenie programu:**

**Obrázok, na ktorom je text

Automaticky generovaný popis**

**Pamäť čistá!**

**Obrázok, na ktorom je text

Automaticky generovaný popis**