**Algoritmy a údajové štruktúry 1**

**Semestrálna práca**

**Názov práce:**

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

**Vypracoval:**

**Tomáš Staroň**

**Obsah**

[1. Návrh aplikácie z pohľadu použitia údajových štruktúr 3](#_Toc104660074)

[1.1 Načítavanie a parsovanie dát z CSV súborov 3](#_Toc104660075)

[1.1.1 Problémy s diakritikou a duplicitnými názvami obcí 4](#_Toc104660076)

[1.2 Párovanie územných jednotiek a priradenie príslušných dát 5](#_Toc104660077)

[2. UML diagram tried aplikácie 6](#_Toc104660078)

[2.1 Popis použitých tried 7](#_Toc104660079)

[3. Popis výpočtu vyhľadávania, filtrovania a triedenia 9](#_Toc104660080)

[4. Používateľská príručka 10](#_Toc104660081)

[5. Záver 12](#_Toc104660082)

# Návrh aplikácie z pohľadu použitia údajových štruktúr

Návrh údajových štruktúr som sa snažil vymyslieť, tak aby najmä všetko fungovalo správne tak ako má a zároveň boli ukladané dáta dostatočne rýchlo prístupné aj pokiaľ ide o výpis viacerých informácií z rôznych časti údajovej štruktúry. V tejto kapitole postupne prejdem ako som si navrhoval svoje štruktúry a ako som využil ich vlastnosti.

## Načítavanie a parsovanie dát z CSV súborov

Pri načítavaní dát z CSV súborov som premýšľal ako najefektívnejšie čítať dáta. Rozhodol som sa pre čítanie využiť kód z internetu, pretože mi prišlo byť zbytočné vymýšľať niečo nové. Použil som nato kód zo stránky https://java2blog.com/read-csv-file-in-cpp/ . Tento kód mi vložil do štruktúry „**vector<vector<string>>**“(použitie tejto štruktúry som vydiskutoval s pánom učiteľom Vargom cez MS Teams a bolo mi povedané, že pokiaľ je využívaná iba na čítanie dát z CSV súborov, tak ju môžem použiť, ďalej využívam už iba štruktúry mnou vytvorené a vytvorené na cvičeniach) postupne písmenka v podstate do dvojrozmernej štruktúry odkiaľ som si ich potom vyťahoval a skladal postupne informácie, ktoré som potreboval.

Dáta som si postupne zo všetkých CSV súborov vyparsoval a povkladal do rôznych štruktúr tak, aby sa mi s nimi neskôr dobre pracovalo. Čítanie mi zabezpečuje trieda CsvReader.h a v nej mám jednotlivé metódy na čítanie každého CSV súbory s dátami. Nižšie je už zobrazené v akej podobe údajovej štruktúre mi dané metódy vrátili tieto dáta:

Zoznam obci - DoubleLinkedList<LinkedList<std::string>\*>\*

* Pre obce som využil DoubleLinkedList z dôvodu, že sa v ňom môžem podľa indexu pohybovať z oboch strán podľa toho kde to je kratšie a tak by sa mal o niečo znížiť čas trvania neskoršej manipulácie s dátami a vkladanie do konečných štruktúr.

zoznam okresov - LinkedList<LinkedList<std::string>\*>\*

zoznamKrajov - LinkedList<LinkedList<std::string>\*>\*

* Použité LinkedList-y z dôvodu, že okresov je málo, LinkedList vyhovuje

vzdelanie - SortedSequenceTable<std::string, Vzdelanie\*>\*

* Použil som utriedenú sekvenčnú tabuľku nato, aby som mohol jednotlivé údaje o Vzdelaní rovno priraďovať konkrétnym obciam na základe kľúču, ktorým je kód územnej jednotky. V CsvReaderi som si rovno vytváral objekty Vzdelania, ktoré potom už len priradím danej územnej jednotke cez smerníky. To isté platí aj pre okresy, kraje a Slovensko, ktorým budem môcť tým pádom tiež jednoducho vkladať tieto údaje, pretože obsahujú sumárne informácie z obcí, ktoré spadajú do daného územného celku.

vek - SortedSequenceTable<std::string, Vek\*>\*

* Pri veku som uplatnil rovnaké zmýšľane ako pri dátach o vzdelaní.

**Spomenuté výsledné údajové štruktúry samozrejme nie sú konečné. V ďalších krokoch s nimi pracujem ďalej.**

### Problémy s diakritikou a duplicitnými názvami obcí

Za spomenutie určite stojí to, že som mal problémy s diakritikou pri načítavaní dát a následnom výpise. Nepodarilo sa mi nastaviť do programu kódovanie ASCII na UTF-8, aby som dokázal vypisovať znaky s diakritikou. Rozhodol som sa teda pristúpiť k odstráneniu diakritiky z excelovských súborov automatizovaným spôsobom. Využil som nato pokyny na tejto webstránke : https://miroslavpecka.cz/blog/odstraneni-diakritiky-v-excelu

Samozrejme potom nastali určite aj prípady kedy viaceré obce mali rovnaké názvy kvôli takémuto kroku, pretože ako kľúče som chcel neskôr používať kľúče, avšak môj návrh uloženia údajových štruktúr si s týmto bez problému poradil. Overil som si či existujú v jednom okrese 2 obce s rovnakým názvom a zistil som, že takýto prípad neexistuje a tak som s týmto nemal žiadny problém, pretože v mojich tabuľkách sú vždy iba obce z jedného okresu a túto tabuľku má ako atribút už daný okres.

## Párovanie územných jednotiek a priradenie príslušných dát

V predošlej podkapitole sú ukázané údajové štruktúry z ktorých som ďalšom kroku išiel vytvárať konečnú podobu dát. Ako prvé som potreboval priradiť krajským územným jednotkám okresy, ktoré do nich patria a následne všetky obce priradiť okresom do ktorých patria. To som urobil v trojitom cykle kde som postupne zisťoval na základe Kódu územnej jednotky to kam patria.

Počas prehľadávania patričných územných jednotiek som si zakaždým vytvoril nový objekt triedy UzemnaJednotka pomocou parametrického konštruktora kde som mu určil prvotné údaje o nej. Vytvorila sa aj tabuľka v ktorej ukladám územné jednotky, ktoré do nej patria, takže pokiaľ som vytvoril objekt Žilinský kraj, tak sa mu aj vytvorila tabuľka, do ktorej vkladám okresy, ktoré do neho patria. Tabuľka má nasledovnú formu: SortedSequenceTable<std::string, UzemnaJednotka\*>

Kľúč : názov územnej jednotky

Dáta : smerník na územnú jednotku

Ako kľúč v nej používam názov územnej jednotky

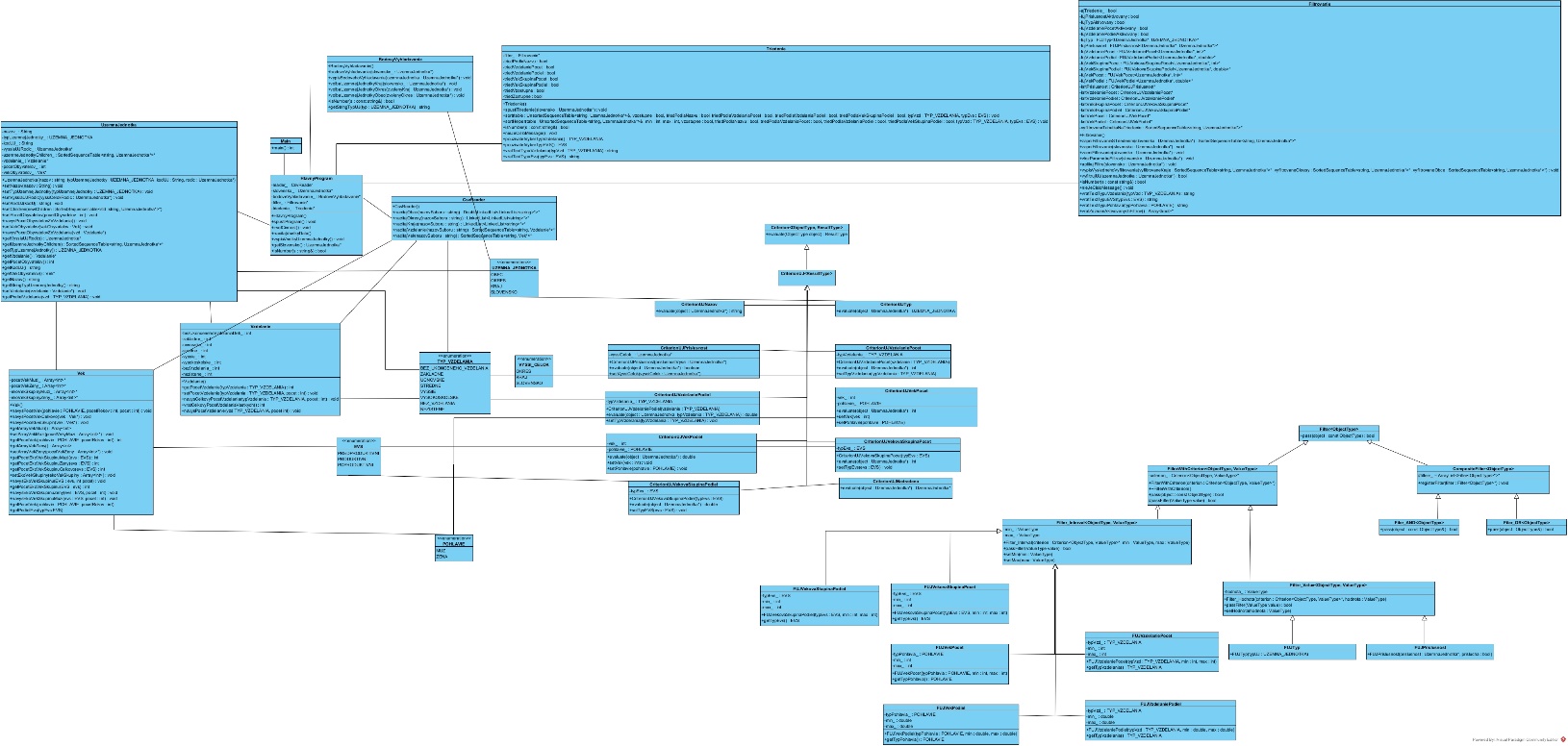
Pokiaľ som našiel postupnú patričnosť a teda najskôr som našiel kraju príslušný okres a následne danému okresu obec, ktorá sa v ňom nachádza, tak som pristúpil k priradeniu smerníkov dát na objekty tried Vzdelanie a Vek. Keďže som sa nachádzal v trojitom cykle a mal som vytiahnuté územné jednotky, ktoré pod seba postupne spadajú, tak som mal uľahčené vytváranie sumárnych údajov pre vzdelanie a vek. Stačilo mi použiť kód obce a použitie metódy find(„kodObce“) a vrátil sa mi už daný objekt Vzdelania alebo Veku. Tieto objekty som priradil najskôr danej obci a následne som tieto objekty využil ako paramtre do metód, ktoré mi zvýšia sumárne dáta v okresoch, krajoch a štáte. Myslím si, že zároveň v kóde to je veľmi prehľadné.

Vo výsledku mám teda objekty územné jednotky, ktoré majú tabuľky svojich potomkov a tvoria teda pomyselnú stromovú štruktúru aj napriek tomu, že sú v tabuľke. Najprv som chcel využiť MultiWayTree, avšak SortedSequenceTable ma presvedčila svojou rýchlosťou. Vytvoril som si aj vo svojej podstate koreň tohto stromu objekt Slovensko od ktorého sa môžem postupne dostať ku ktorejkoľvek územnej jednotke a získať jej dáta. Smerník na objekt Slovensko mám uložený v triede HlavnyProgram odkiaľ riadim chod celej aplikácie. Myslím si, že môj návrh je veľmi dobrý, pretože som pri práci nemal žiadne problémy.

1. Premenovanie názvov obcí, ktoré majú svojho menovca

Po priradení dát som si ešte upravil názvy obcí tak, aby pri vypisovaní nebol používateľ zmätený pokiaľ sa mu zobrazia na obrazovke napríklad 2 obce s názvom Dubrava. Všetkým obciam som priradil za ich meno do zátvoriek ešte názov okresu v ktorom sa daná obec nachádza. Kľúče do tabuliek som meniť nemusel, takže vyhľadávanie obcí sa mi nijako neskomplikovalo.(Názvy som menil iba vo vnútri objektu v rámci atribútu „nazovUJ - string“).

# UML diagram tried aplikácie

Pre veľkosť, ktorú má môj UML diagram tried odporúčam otvoriť si vyexportovaný obrázok diagramu z programu Visual Paradigm v ktorom som ho vytváral. Uložený je v adresári s názvom „UML“ v rámci odovzdaného súboru.

## 2.1 Popis použitých tried

* Main.cpp – táto trieda funguje ako hlavná trieda na spustenie aplikácie, obsahuje smerník na objekt triedy HlavnyProgram a inicializuje heap\_monitor na kontrolu memory leakov.
* **HlavnyProgram.h** – riadi hlavný chod aplikácie, vyvolá načítanie dát z inej triedy, poskytuje možnosti používateľovi na používanie aplikácie a stará sa o najhlavnejšiu činnosť. Obsahuje smerník na objekt Slovensko, od ktorého sa dokážeme dostať postupne ku každej územnej jednotke a ich dátam. Objekt Slovensko používam ako parameter do bodového vyhľadávania, filtrovania a triedenia, keďže to je objekt od ktorého sa dostávam v aplikácií ku všetkým potrebným dátam.
* **CsvReader.h** – poskytuje metódy, ktoré dokážu prečítať a naparsovať potrebné dáta o územných jednotkách, vzdelaní a veku z dát, ktoré máme k dispozícií. Má metódy prispôsobené na konkrétny .csv súbor. Vracajú objekty v ktorých sú do už spomenutej formy dočasne uložené dáta na neskoršie spracovanie.
* **UzemnaJednotka.h –** inštancie tejto triedy sú územné jednotky. Hlavné atribúty sú kód územnej jednotky, názov, počet obyvateľov, smerník na objekt vyššej územnej jednotky. Medzi dôležitejšie atribúty patrí atribút „uzemneJednotkyChildren\_“, ktorý je tabuľkou, v ktorej sú uložené smerníky na nižšie územné jednotky, ktoré pod ňu patria. Informácie o veku a o vzdelaní ukladám cez smerníky na objekty tried Vek a Vzdelanie.
* **Vek.h** – trieda ma za úlohu ukladať informácie o veku pre danú územnú jednotku. Ukladám ich v štruktúre Array kde mám uložené veky pre jednotlivé veky mužov, žien a ukladám aj takisto počty ekonomických vekových skupín.
* **Vzdelanie.h** – obsahuje atribúty obyčajných Integerov, ktoré ukladajú počty ľudí, ktoré majú dané vzdelanie. Smerníky na objekty tejto triedy sú ukladané do objektov UzemnaJednotka
* **Enum UZEMNA\_Jednotka –** OBEC, OKRES, KRAJ, SLOVENSKO
* **Enum VYSSI\_CELOK -** OKRES, KRAJ, SLOVENSKO
* **Enum Pohlavie** – trieda obsahuje enumy pre pohlavie – MUZ, ZENA, VSETCI
* **Enum TYP\_VZDELANIA –** BEZ\_UKONCENEHO\_VZDELANIA\_DETI, ZAKLADNE, UCNOVSKE, STREDNE, VYSSIE, VYSOKOSKOLSKE, BEZ\_VZDELANIA, NEZISTENE
* **Enum EVS –** PREDPRODUKTIVNI, PRODUKTIVNI, POPRODUKTIVNI
* **BodoveVyhladavanie .h–** táto trieda mi poskytuje metódy, pomocou ktorých si dokáže používateľ zvoliť presnú územnú jednotku o ktorej chce vyhľadať základné informácie. Objekt tejto triedy mám vytvorený ako atribút v objekte HlavnyProgram. Bodové vyhľadávanie spúšťam pomocou metódy void bodoveVyhladavanie(slovensko : UzemnaJednotka\*).
* **Filtrovanie.h** – ako hovorí názov, tak táto trieda poskytuje filtrovanie územných jednotiek, pri spustení filtrovania ide do parametru objekt Slovensko od ktorého pomocou mnou vytvorených algoritmov filtrujem dané územné jednotky a postupne ich prechádzam či vyhovujú daným filtrom. Používateľ si zvoli filtre aké chce a tak isto aj ich parametre.
* **Triedenie.h –** Triedenie má na starosti utriediť výsledné prefiltrované územné jednotky. Používateľ môže zvoliť kritérium podľa ktorého chce utriediť dané územné jednotky a či ich chce usporiadať vzostupne alebo zostupne.
* Obrázok, na ktorom je text

  Automaticky generovaný popis**Kritéria –** základ pre moje kritéria mám zo zadania v ktorom presne definované čo majú robiť, aké majú mať výstupné hodnoty a podobne.. Keďže boli všetky robené presne podľa zadania, tak sem prikladám screenshoty popisov kritérií

Obrázok, na ktorom je text

Automaticky generovaný popis

**Obrázok, na ktorom je stôl

Automaticky generovaný popisObrázok, na ktorom je text

Automaticky generovaný popis**

**Obrázok, na ktorom je text

Automaticky generovaný popis**

* **Filtre –** Filtre som implementoval tak isto podľa príručky a teda vytvoril som si jeden hlavný od ktorého mám potomkov CompositeFilter a FilterWithCriterion. Vo svojej práci nakoneic ani nevyužívam CompositeFilter a jeho potomkov, avšak mám ich v diagrame a takisto aj v práci, avšak nie použité.

Od triedy FilterWithCriterion ďalej idú potomkovia Filter\_Interval a Filter\_Value a ako napovedajú názvy, tak filtrujú jeden podľa intervali hodnôt min/max a druhý podľa konkrétnej presnej hodnoty.

Od Filter\_Value mám potomkov FUJTyp a FUJPrislusnost, tieto 2 filtre vracajú bool hodnotu podľa toho či daná územná jednotka je daného typu a či je v príslušnosti danej územnej jednotky.

Od Filter\_Interval mám Filtre: FUJVekovaSkupinaPocet, FUJVekovaSkupinaPodiel, FUJVekPocet, FUJVekPodiel, FUJVzdelaniePocet a FUJVzdelaniePodiel.

# Popis výpočtu vyhľadávania, filtrovania a triedenia

n – počet prvkov , k – počet vypisovaných údajov

**Načítavanie dát:**

Pri načítavaní dát musím prejsť cez všetky riadky CSV súborov. Znaky z tých riadkov si ukladám do vektora. Následne mi vznikne 2D matica zo znakov. Tieto znaky ďalej spracujem do samostatných riadkov a ďalej ich spracúvam už subjektívne od konkrétneho CSV súboru. Výsledné štruktúry z každého súboru som popísal v predošlých kapitolách, nebudem sa tu k tomu teda vraciať. Popíšem už iba konečné spájanie údajov dokopy.

Používam trojitý cyklus v podstate „for each“ kde mám použité pre kraje LinkedList, pre okresy LinkedList a pre obce DoubleLinkedList(aby to bolo rýchlejšie kedže ich je viac.

Výsledná zložitosť: O(n\*n\*n/2)

Následne sa táto zložitosť ešte vynásobí pri načítavaní Objektov Veku a Vzdelania z SortedSequenceTables. Kde je zložitosť O(log n + log n)

**Výsledná zložitosť by teda mala byť O(n\*n\*n/2 + log n + log n)**

**Bodové vyhľadávanie:**

* Vyhľadanie informácií o konkrétnom kraji – O( [log n] + k)
* Vyhľadanie informácií o konkrétnom okrese– O( 2\*[log n] + k)
* Vyhľadanie informácií o konkrétnej obci– O( 3\*[log n] + k)

**Filtrovanie :**

Pri filtroch je nevýhodou, že nato ak chcem zistiť či daná územná jednotka vyhovuje filtru tak ju musím preveriť daných kritériom filtra, z toho dôvodu majú všetky filtre zložitosť O(n). Avšak pri procese filtrovania to aj tak nevidno a aplikácia ide veľmi svižne.

**Triedenie :**

Pri triedení poskytujem na výber triediace kritéria podľa ktorých sa triedi pomocou algoritmu, ktorý sedí pseudokódu QuickSort-u(zložitosť O(N\*log2N)). Pokiaľ je kritérium práve kľúč územnej jednotky, tak sa vyberie iba kľúč, niekedy môže byť kritériom triedenia aj rôzny atribút alebo podobne niečo čo treba ešte vypočítať, preto zadefinujem konštantu **„h“** ktorá označuje zložitosť vypočítania danej hodnoty potrebnej pre triedenie.

**Výsledná zložitosť triedenia: O(N\*log2N + h)**

# Používateľská príručka

Po spustení programu sa najskôr načítajú všetky potrebné dáta do aplikácie, v okne aplikácie sa medzitým zobrazuje informačný výpis o tom aký proces momentálne prebieha.

Keď sú už dáta načítané v aplikácií sa otvorí **Hlavné menu**, kde si používateľ môže vybrať činnosť, ktorú chce vykonať. Svoju možnosť označí pomocou čísiel, ktoré sa nachádzajú pri danej operácií.

Aplikácia vo väčšine prípadoch očakáva, že používateľ sa môže aj pomýliť a tak ho iba upozorní na neexistujúcu voľbu a ponúkne mu opätovný výber.

V hlavnom menu sa dá aplikácia samozrejme aj ukončiť.

Aplikácia sa ovláda pomocou čísiel, ktoré sú jasne označené pre „akcie“ v aplikácií.

**Bodové Vyhľadávanie:**

Po vybratí bodového vyhľadávania používateľ následne volí, ktorú územnú jednotku chce nájsť a vypísať o nej informácie. Robí tak postupne a to tým spôsobom, že najskôr zvolí kraj v ktorom je obec, potom okres a následne zvolí obec ktorú požaduje.

Pokiaľ chce používateľ vypísať informácie o kraji, tak stačí, aby ho najskôr zvolil a potom pri výbere okresov zvolil možnosť pod číslom „0“ a vypíšu sa mu informácie o požadovanom kraji.

**Filtrovanie:**

Pri filtrovaní používateľ najskôr zadá, ktoré z filtrov chce aplikovať na danú množinu územných jednotiek. Môže ich zapnúť/vypnúť. Keď je so zoznamom zapnutých filtrov spokojný, tak potvrdí svoj výber zadaním čísla „0“.

Následne je od používateľa potrebné postupné vkladanie parametrov filtrov. Vždy sa pri danom požadovanom parametri nachádza príslušný popis možností voľby hodnoty parametra filtra a takisto nevadí pokiaľ zadá neaplikovateľnú hodnotu, bude nato upozornený a môže voľbu vykonať opäť.

Po výber všetkých parametrov sa filtre aplikujú a okamžite vypíšu územné jednotky, ktoré dané filtre spĺňajú a vypíšu sa hodnoty týchto územných jednotiek pre dané kritéria použitých filtrov.

**Triedenie:**

Triedenie má zo začiatku rovnaký spôsob ako filtrovanie. Takže používateľ najskôr zapne požadované filtre, zvolí ich parametre a vyfiltrujú sa územné jednotky a vytvorí sa nezoradení zoznam. Následne je používateľ vyzvaný nato, ktoré kritérium chce použiť na triedenie, pokiaľ vyberie kritérium, ktoré potrebuje dodatočné parametre, bude vyzvaný na ich zadanie. Následne sa aplikácia opýta už iba či chce používateľ utriediť územné jednotky podľa daného kritéria vzostupne alebo zostupne. Po tejto voľbe sa na pozadí územné jednotky už rýchlo utriedia a vypíšu sa na obrazovku.

# Záver

Výsledná aplikácia je použiteľná a sám som ju otestoval. Testovanie aplikácie po úplnom skončení programovania, ktoré som vykonal trvalo približne 3 hodiny. Za tento čas som sa pokúšal o vytvorenie najrôznejších možných scenárov, ktoré by mohli nastať pri používaní aplikácie. Vždy som skontroloval či je pamäť po ukončení programu čistá a neostali v nej nejaké zbytky.

Keby som mal viac času, určite by som sa viacej posnažil ešte s user interface-om, avšak myslím si, že v súčasnej forme sa dá tiež veľmi intuitívne a jednoducho používať.

Pri práci som pracoval samostatne a musím povedať, že mi táto semestrálna práca dala veľa. Naučil som sa ako pracovať s jazykom C++ a najmä ako správne využívať možnosti, ktoré nám tento jazyk prináša.

**Údajové štruktúry, ktoré sú použité v práci sú všetky, ktoré sa naprogramovali na cvičeniach alebo som si ich naprogramoval sám. Jediná výnimka je pri spomenutom čítaní dát z CSV súborov, avšak túto tému som prediskutoval s pánom učiteľom Vargom a bolo mi povedané, že pokiaľ to je využité iba na čítanie z CSV súborov, tak to nevadí.**