Žilinská Univerzita v Žiline, Fakulta Riadenia a Informatiky

**Algoritmy a údajové štruktúry 1**

Semestrálna práca 2

2018/2019 Jozef Kubík

5ZY022

# Návrh aplikácie

## Použité údajové štruktúry

* Array
* UnsortedSequenceTable

Prípady a dôvody použitia práve týchto údajových štruktúr:

**Array** – použité na uchovávanie smerníkov na objekty *Kraj* a *Okres* v triede *System*

*Dôvod použitia poľa:* počty krajov aj okresov sú známe vopred a počas behu programu sa nemenia, zároveň ich nepotrebujeme žiadnym spôsobom zoraďovať a naopak – predovšetkým pri načítavaní je častý prístup na daný prvok podľa indexu

**UnsortedSequenceTable** – použité na uchovávanie smerníkov na objekty *Obec* v triede *System*

*Dôvod použitia neutriedenej sekvenčnej tabuľky:* obce je potrebné v semestrálnej práce triediť pomocou triediaceho algoritmu (ja som zvolil *ShellSort*), ktorý “spolupracuje“ práve s neutriedenou sekvenčnou tabuľkou, takže výber bol vopred jasný

Neutriedená sekvenčná tabuľka je “vystavaná“ na *ArrayList-e* a ten nad *Array*; využil som faktu, že je vopred známe, koľko Obcí je celkovo na Slovensku a do spomenutých štruktúr som pridal konštruktory, ktoré zabezpečia, že pole je pri vytváraní dostatočne veľké a počas načítavania prvkov sa nemusí realokovať. Z toho plynie časová úspora.

## Použité algoritmy

* ShellSort

Dôvody použitia a spôsob implementácie:

**ShellSort** – keďže bolo nevyhnutné zabezpečiť, aby horná asymptotická zložitosť algoritmu pre zoradenie bola nižšia ako O(n2), siahol som po vylepšených metódach triedenia, medzi ktoré patrí aj ShellSort, ktorý dosahuje zložitosť O(n1.2)

*Implementácia:* ShellSort je algoritmus v ktorom sa zmenšuje krok medzi prvkami, ktoré porovnáva až dosiahne 1 (tj. porovnáva susedné prvky). V mojej implementácii som za začiatočný krok zvolil n/2.2 a následne ho zmenšoval rovnakým spôsobom (aktuálny krok/2.2) až kým nedosiahol hodnotu 1, pričom ak mal byť krok 2, hneď bola zvolená hodnota 1. Tento spôsob, kde sú kroky algoritmu z celočíselného radu násobkov čísla 2.2 (začínajúcich od 1) by mal zabezpečiť najmenšiu možnú dosiahnuteľnú zložitosť ShellSort-u.

## Zložitosti všetkých operácií

Kde:

k – počet krajov,

m – počet okresov,

n – počet obcí

Poznámka – funkcionalita č.1 je implementovaná v bode 3 a funkcionalita č. 2 v bode 4

**0. Načítavanie pri spustení** – výsledná zložitosť: **O(k + m + n)**

Pri spustení sa vytvorí potomok triedy *System*, ktorý obsahuje pole o dostatočnej veľkosti pre všetky kraje, pre všetky okresy a neutriedenú sekvenčnú tabuľku pre obce.

Naplnenie poľa má zložitosť O(k), resp. O(m), pretože pri vytváraní potomkov triedy *Kraj*, resp. *Okres* (aj *Obec*) uvažujeme zložitosť O(1), za predpokladu, že sa nebudú pridávať počty kôl prezidentských volieb (inak by zložitosť bola O(p) pre každý objekt, kde p – počet kôl)

Naplnenie neutriedenej sekvenčnej tabuľky n prvkami má zložitosť O(n), ak zanedbáme rozširovanie poľa, na ktorom je tabuľka postavená (toto môžeme urobiť z dôvodu popísaného v časti použité údajové štruktúry). Taktiež musíme zanedbať prehľadávanie celej tabuľky pri vkladaní nového prvku (kontrola unikátnosti), čo môžeme urobiť, keďže máme zabezpečené, že dáta použité ako kľúč *sú unikátne* (z toho dôvodu bola popísaná kontrola v štruktúre zmazaná).

**3.a) Vypísanie informácií o územných jednotkách** – výsledná zložitosť: **O(k) alebo O(m) alebo O(n)**

Pri filtrovaní musíme prejsť všetkými prvkami buď Krajov, Okresov alebo Obcí – toto si užívateľ zvolí na začiatku, skontrolovať hodnotu ich atribútu s hľadaným kritériom O(1). Následné vypísanie požadovaných výstupov má zložitosť O(1) (za predpokladu, že sa nebude zvyšovať počet kôl volieb).

**3.b) c) Vypísanie informácií o územných jednotkách** – výsledná zložitosť: **O(k + m +n)**

Rovnako ako v bode vyššie, kontrola atribútu objektu s hľadaným zadaným kritériom má zložitosť O(1) a výpis O(1). Rozdiel je v tom, že v tomto prípade (pri filtroch voliči a účasť) musíme prejsť všetkými 3 údajovými štruktúrami, v ktorých máme uložené *Obce, Okresy, Kraje*.

**4. a) b) c) Zoradenie územných jednotiek splňujúcich daný filter** – výsledná zložitosť:

**O(k + n + N + N1.2 + 2N) alebo O(m + n + N + N1.2 + 2N)**

kde N – počet obcí splňujúcich oba filtre

Najskôr sa prejde buď pole Okresov alebo pole Krajov (podľa voľby používateľa), aby sme získali smerník na danú vyššiu územnú jednotku, toto má zložitosť O(k) alebo O(m).

Následne prechádzame všetkými obcami O(n), pričom podmienky pre oba filtre kontrolujeme naraz (či patrí k územnej jednotke a či splňuje zadanú účasť), čiže zložitosť je O(1) pre oba.

Hneď, ak spĺňajú filtre, ich priraďujeme aj do novej neutriedenej tabuľky O(N), keďže kontrola unikátnosti nie je potrebná.

Následne ich utriedime algoritmom ShellSort, ktorý má udávanú zložitosť O(N1.2).

Po ich utriedení, ich vypíšeme raz zostupne O(N), potom zostupne O(N).

# Diagram tried

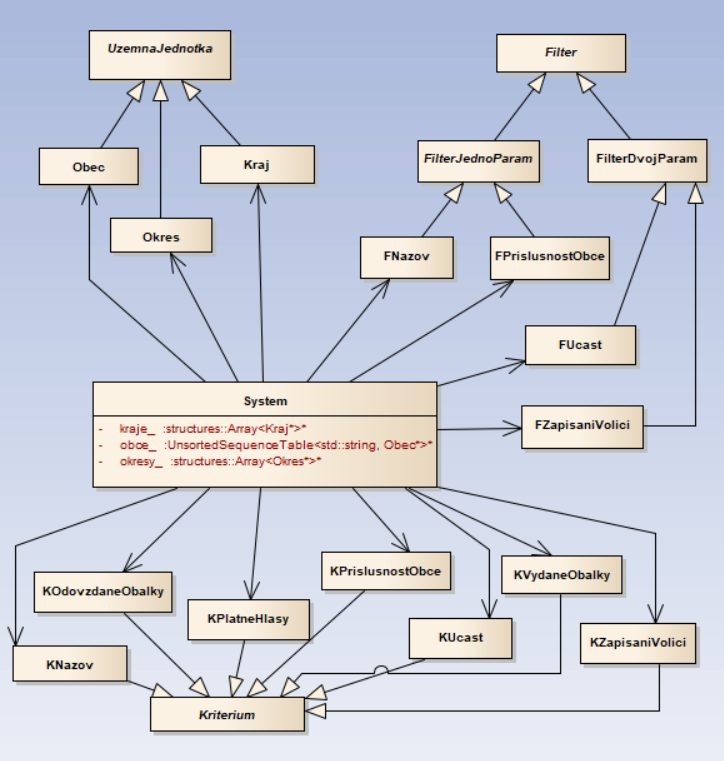


Diagram najdôležitejších súčastí systému – triedu *System* a jej súčasti, kritéria, filtre a triedy pre územné jednotky

# Používateľská príručka

Po spustení aplikácie sa načítajú všetky potrebné údaje zo súboru, čo môže, pri najhoršom, pár sekúnd zabrať.

Následne užívateľ obsluhuje program zadávaním číslic 0 až 9, v prípade desatinných čísiel sa používa desatinná bodka ‘**.**‘ (nie čiarka – ‘**,**‘). Na potvrdenie vstupu slúži klávesa Enter. Ovládanie je veľmi intuitívne a užívateľ vždy dostane hlásenie, aký vstup od neho aplikácia očakáva.