**ŽILINSKÁ UNIVERZITA V ŽILINE**

FAKULTA RIADENIA A INFORMATIKY

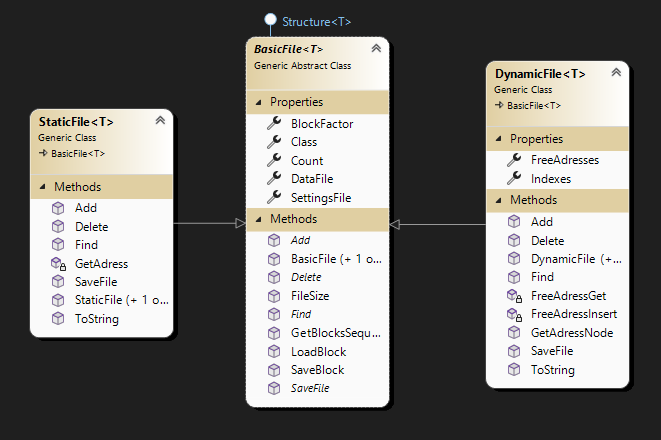
Algoritmy a údajové štruktúry 2

MATEJ MAŽGÚT

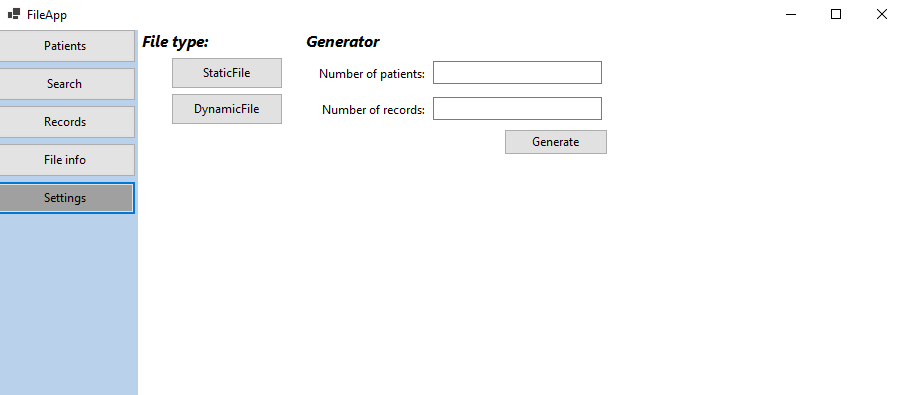
**Elektronická zdravotná karta**

# Použité údajové štruktúry

V projekte budem využívať mnou implantované štruktúry statického a dynamického hešovacieho súboru. Podrobnejší popis štruktúr je zobrazený na UML diagrame nižšie.



Pri spustení aplikácii si používateľ môže vybrať ci chce pracovať s dynamickým alebo statickým typom hešovacieho súboru. Následne si môže vybrať či chce pokračovať s predošlými dátami alebo chce vytvoriť nový súbor. Po zvolení súboru je možné vygenerovať dáta a to zadaním celkového počtu pacientov a hospitalizácii.



Ďalej sa tu nachádza záložka pacient, kde je možné pridať alebo odstrániť pacienta. V záložke search je možné vyhľadávať pacienta podľa rodného čísla, prípadne aj záznam zadaním id záznamu.

V záložke record je možné pridať nový záznam, ukončiť exitujúci alebo vymazať záznam. Posledné tlačidlo File info slúži na sekvenčný výpis celého súboru.

## Ukladanie štruktúr

Túto funkcionalitu zabezpečuje metóda SaveFile(). Pri statickom súbore si do pomocného súboru uložím blok faktor pôvodného súboru a tiež počet blokov. Pri dynamickom si ukladám okrem blok faktoru aj štruktúru s voľnými adresami a všetky externé vrcholy dynamického stromu. Pre každý externý vrchol si ukladám jeho index v strome a ak ma priradený blok v súbore tak aj adresu na daný blok a počet prvkov v bloku. Tieto informácie si ukladám do separátneho textového súboru, ktorý pri možnosti pokračovania s predošlými dátami načítam a vybudujem pôvodný súbor.

Pri statickom súbore to znamená že pri vytváraní súboru určím pôvodný blok faktor a ďalej budem pracovať so súborom s predošlými dátami rovnako.

V dynamickom súbore musím tiež najskôr určiť blok faktor, následne pridám voľné adresy do zoznamu a ako posledné vybudujem nanovo strom s indexami v operačnej pamäti. To robím tak, že postupne čítam externé vrcholy z textového súboru a pridávam ich do stromu. Postupujem od korena stromu a postupne pridávam interné vrcholy až pokým nemôžem pridať načítaný externý. Potom mu priradím adresu zo súboru a počet vládnych blokov v súbore.

## Manažér voľných blokov v dynamickom súbore

Manažér v mojom prípade pracuje so štruktúrou utriedeného zreťazeného zoznamu. Keď sa uvoľni blok v štruktúre adresa sa vloží do zoznamu na správne miesto. Skontroluje sa či adresa nie je na konci súboru, ak áno skráti daný súbor a a tá istá operácia sa vykoná na predchodcovi v zozname, až kým nenarazí na predchodcu, ktorý sa nenachádza na konci súboru. Pri vytvorení nového bloku v súbore a teda priradení novej adresy si manažér skontroluje ci nie je zoznam s adresami prázdny, ak nie je vyberie sa adresa zo začiatku utriedeného zoznamu. Inač sa priradí nová adresa a súbor sa zväčši.

## Zložitosti operácií

V štruktúre je možné vykonať tri operácie a to vloženie, vymazanie a nájdenie.

## Operácia hľadania

Statická štruktúra pristupuje do súboru pri hľadaní iba jeden krát. Získa hash z objektu, ktorý chce vložiť. Následne načíta blok s daným hashom a vráti hľadaný objekt z bloku.

Pri dynamickom súbore prebieha hľadanie rovnako a tak je počet prístupov do súboru rovný jednej.

## Operácia vloženia

Statický súbor pri vložení nového objektu pristupuje do súboru dva krát. Najskôr načíta blok na adrese hashu daného objektu. Potom do bloku pridá vkladaný objekt a pristupuje do súboru druhý krát, aby zmenený blok do súboru zapísal. Problém tu nastáva ak je dańy blok plný, vtedy sa nový objekt do súboru nepridá.

Dynamický súbor umožní vložiť nový objekt aj keď je blok plný. Pri voľnom mieste v bloku prebieha vloženie rovnako ako pri statickom a tak počet prístupov do súboru je dva. Pri plnom bloku sa blok zo súboru načíta, vytvorí sa nový a objekty sa podľa hashu rozdelia do blokov. Následne sa oba bloky zapíšu do súboru, ak nie sú prázdne. Táto operácia sa opakuje až kým sa v jednom z blokov nevytvorí miesto pre nový objekt. Dynamický súbor teda pristúpi minimálne dva krát do súboru pri vkladný nového objektu.

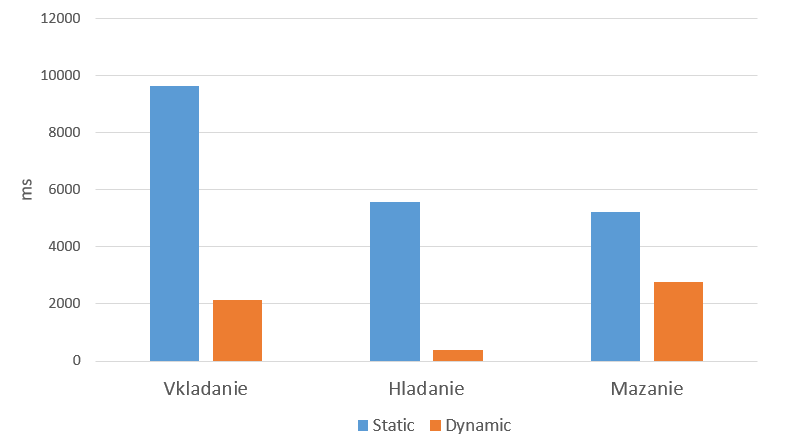
## Operácia mazania

Operácia mazania v statickom súbore pristupuje do súboru dva krát. Prvý krát, aby blok v ktorom je mazaný objekt načítala a druhý, aby blok s vymazaným objektom zapísala do súboru.

Pri dynamickom súbore neviem určiť presný počet prístupov do súboru, lebo môže dôjsť k reorganizácii blokov. Minimálny počet prístupov, ale môže byť nulový. Ten nastáva v prípade že sa v bloku nachádza iba mazaný objekt a tak sa zruší celý blok. Potom tu je prípad, keď je v bloku viac objektov ako jeden a vtedy sa mazanie vykoná rovnako ako pri statickom súbore. Následne je ale nutné skontrolovať susedný vrchol v indexovej štruktúre v operačnej pamäti a ak je možne zlúčiť ich do jedného bloku. Táto operácia musí pristúpiť do súboru znovu a načítať susedný blok . Prvky sa zlúčia do jedného bloku a ten pristúpi do súboru a uloží sa.

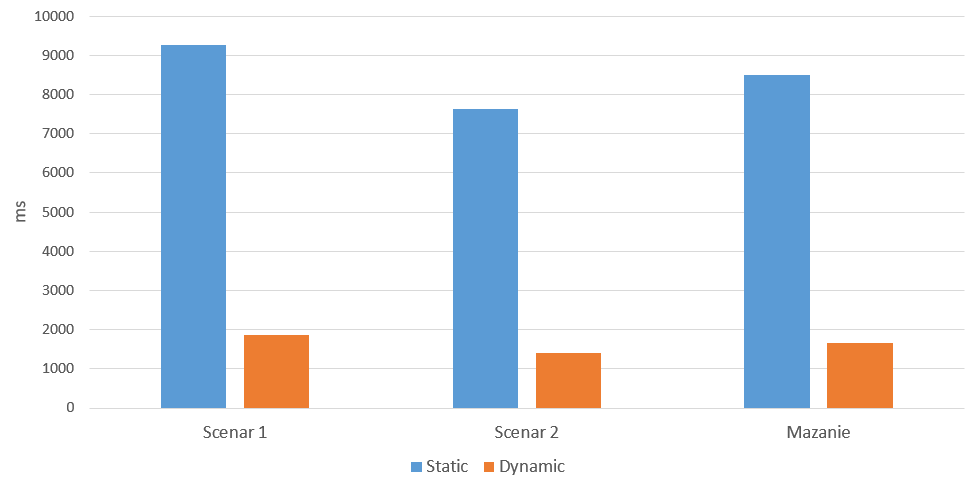
## Porovnanie rýchlosti

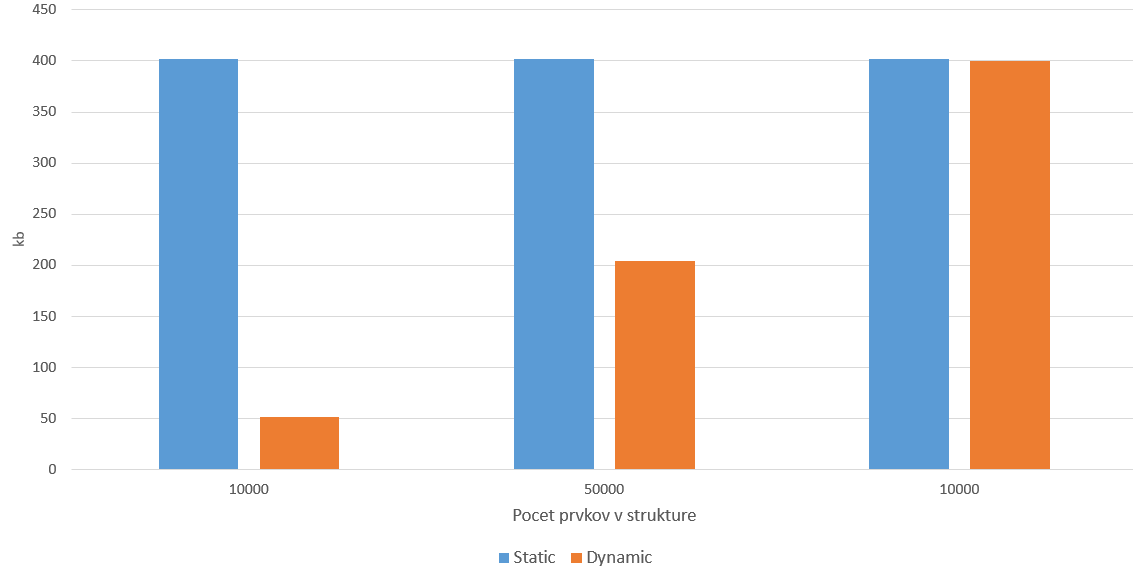
Pri porovnávaní som využíval identickú množinu dát kladných čísel od 1 po 100 000. Aby sa všetky dáta zmestili do statického súbor, veľkosť blok faktoru pri statickom súbore som zvoli ako sqrt(n), kde n je počet vkladaných dát. Statický súbor mal teda blok faktor 320. Pri dynamickom súbore som zvolil blok faktor 50, keďže pri preplnení bloku sa vytvorí nový a štruktúra tak nie je limitovaná blok faktorom. Menší blok faktor zabezpečí čítanie a zapisovanie menšieho množstva dát pri prístupe do súboru.

Následne som vykonal testy vloženia, mazania a hľadania 100 000 prvkov. Pre každú operáciu som vykonal 50 opakovaní a spriemeroval výsledné časy operácie. Výsledky porovnania sú v nasledujúcom grafe.

Na základe testov vieme povedať, že rýchlejšia štruktúra bola dynamický hashovaci súbor. Hlavnú úlohu v tom hral blok faktor, keďže pri jednotlivých operáciách musím načítať väčšiu množinu objektov zo súboru.

Ďalej som vykonal test kde sa operácie striedali. Na začiatku štruktúra obsahovala 40 000 prvkov a vykonal som tri scenáre testovania. V prvom bol pomer operácii 60% - vkladania , 20% - mazania, 20% - hľadania. Pre obe štruktúry som vykonal 10 opakovaní. V druhom bol pomer operácii 40% - vkladania , 40% - mazania, 20% - hľadania a tretom 40% - vkladania , 20% - mazania, 40% - hľadania. Výsledky sú zobrazené na nasledujúcom grafe.

Pri každom zo scenárov sa preukázala taktiež väčšia rýchlosť dynamického hashsovacieho súboru.

Ďalej môžete vidieť porovnanie pamäťovej náročnosti statického a dynamického súboru pri rôznom počte prvkov, kde blok faktor statického súboru je 320 a dynamického je 50.

Staticý súbor ma rovnakú veľkosť počas celej jeho existencie. V dynamicom sa velkost mení podľa počtu prvkov. Čim je prvkov viac tým sa približuje aj jeho veľkosť, veľkosti statického súboru.