ŽILINSKÁ UNIVERZITAV ŽILINE
Fakulta riadenia a informatiky

Semestrálna práca

Algoritmy a údajové štruktúry 1 5UI124

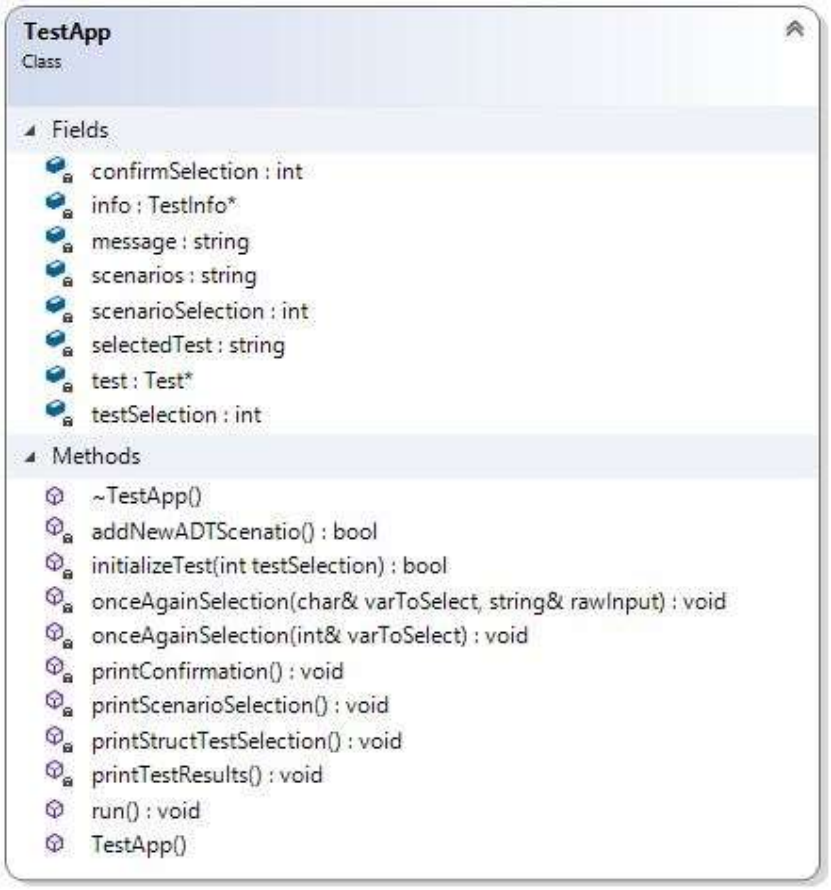
2020/2021 Adam Lány 5ZYI33

Návrh štruktúry testov a testovacej aplikácie

Testovacia aplikácia je reprezentovaná konzolovou aplikáciu, kde si používateľ môže zvoliť pre aký typ dátovej štruktúry budú vykonané testy a následne si môže vybrať aj z testovacích prípadov pridelených k jednotlivým typom štruktúr (pri ADT zoznamoch je možnosť vytvoriť si vlastný scenár). Po prebehnutí testov sa používateľovi zobrazia informácie o vykonaných testoch a aplikácia sa ukončí. Počas testovania je vygenerovaný príslušný CSV súbor v do priečinka DATA_CSV.

TestApp

Trieda ktorá predstavuje testovaciu aplikáciu, komunikuje priamo s používateľom a umožňuje mu výber testovacích scenárov pomocou konzolového vstupu a výstupu na obrazovku. Stará sa o inicializovanie testu, výber testovacieho scenáru z možností pre daný test štruktúry a následne jeho spustenie. Po ukončení testu trieda vypíše informácie (celkový čas testu, počet vykonaných operácií) na obrazovku používateľa.



void TestApp::run()

Metóda pre spustenie celej testovacej aplikácie. Zabezpečuje všetky potrebné výpisy na konzolový výstup pre používateľa. V jej vnútri sa volajú ostatné súkromné pomocné metódy. Ako prvá sa volá pomocná metóda pre výber testu pre danú štruktúru. Ak používateľ zadá platný vstup inicializuje sa testovacia trieda pre danú štruktúru a zavolá sa pomocná metóda pre výpis výberu jednotlivých testovacích scenárov. Jednotlivé možnosti výberu týchto scenárov sa získajú z inicializovanej triedy pre test štruktúry. Každý test štruktúry má iné možnosti výberu scenárov. Po vybratí scenáru sa aktualizuje notifikácia so správou s vybraným testom štruktúry a vybraným testovacím scenárom v ľavom hornom rohu konzoly a zobrazí sa hláška pre potvrdenie výberu testu a scenáru. Ak používateľ tento výber nepotvrdí aplikácia sa ukončí. V opačnom prípade sa spustí test pre danú štruktúru, kde sa pomocou parametra zvolí testovací scenár, ktorý sa má spustiť. Druhým parametrom, ktorý sa odovzdáva referenciou je objekt, ktorý drží v sebe potrebné informácie o teste. Po ukončení testu sa tieto informácie vypíšu pomocou metódy na konzolový výstup.

void TestApp::printStructTestSelection()

Pomocná metóda vykresľuje na konzolový výstup selekciu výberu testov pre jednotlivé štruktúry. Výpis sa realizuje pomocou `cout` objektu. Po výpise možností nasleduje výzva pre zadanie výberu používateľa. Používateľský vstup je realizovaný pomocou objektu `cin`. V prípade. že používateľ zadá akýmkoľvek spôsobom neplatný vstup, vypíše sa výzva pre opätovné zadanie vstupu.

void TestApp::printScenarioSelection()

Pomocná metóda vykresľuje na konzolový výstup selekciu výberu testovacích scenárov pre vybraný test štruktúry. Výpis sa realizuje pomocou `cout` objektu.Po výpise možností nasleduje výzva pre zadanie výberu používateľa. Používateľský vstup je realizovaný pomocou objektu `cin`. V prípade. že používateľ zadá akýmkoľvek spôsobom neplatný vstup, vypíše sa výzva pre opätovné zadanie vstupu. **Pri výbere testu pre štruktúru ADT zoznam sa vypíše, mimo testovacích scenárov, aj možnosť pre vytvorenie vlastného scenáru.

bool TestApp::addNewADTScenatio()

Metóda vytvorí a uloží nový scenár pre ADT zoznam. Používateľovi vypíše výzvy pre zadanie pomerov jednotlivých skupín operácií a uloží ich do CSV súboru. Súčet jednotlivých pomerov musí vždy dávať 100 aby bol tento scenár úspešne uložený. V opačnom prípade bude vypísaná hláška o neúspech uloženia v ľavom hornom rohu konzolového výstupu. Po úspešnom alebo neúspešnom uložení vlastného scenáru je používateľovi zonova zobrazený výber testovacích scenárov. Ak bol vlastný scenár úspešne uložený, tak je jeho výber zobrazený v tomto zozname scenárov. Pre uloženie scenáru do CSV súboru je použitá trieda `FileOutputHander`, kde sa nastaví názov príslušného súbore pre zápis a že chcem pridávať na koniec tohto súboru. pomocou metódy tejto triedy sa predá už hotový riadok pre zápis do CSV.

void TestApp::printConfirmation()

Pomocná metóda pre potvrdenie výberu daného testu štruktúry a scenáru. Ak používateľ tento výber nepotvrdí aplikácia sa ukončí. V opačnom prípade pokračuje ďalej.

void TestApp::printTestResults()

Metóda zobrazí na konzolový výstup informácie z objektu Info o vykonanom teste.

void TestApp::onceAgainSelection(int& varToSelect)

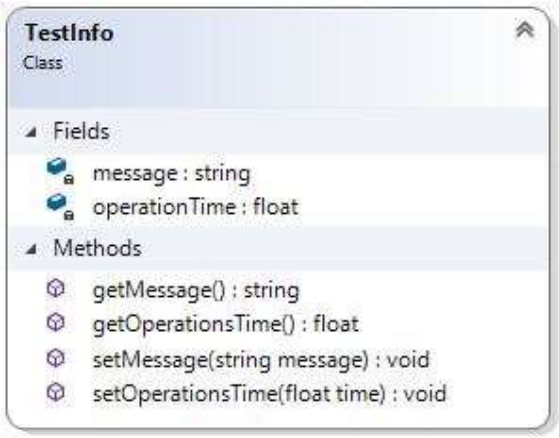
Metóda riadenie opätovného vstupu používateľa. Výber používateľa je riadený pomocou číselnej premennej (používateľ zadáva číslo ako svoj výber).

void TestApp::onceAgainSelection(std::string& rawInput)

Metóda riadenie opätovného vstupu používateľa. Výber používateľa je riadený pomocou string premennej (používateľ zadáva string ako svoj výber).

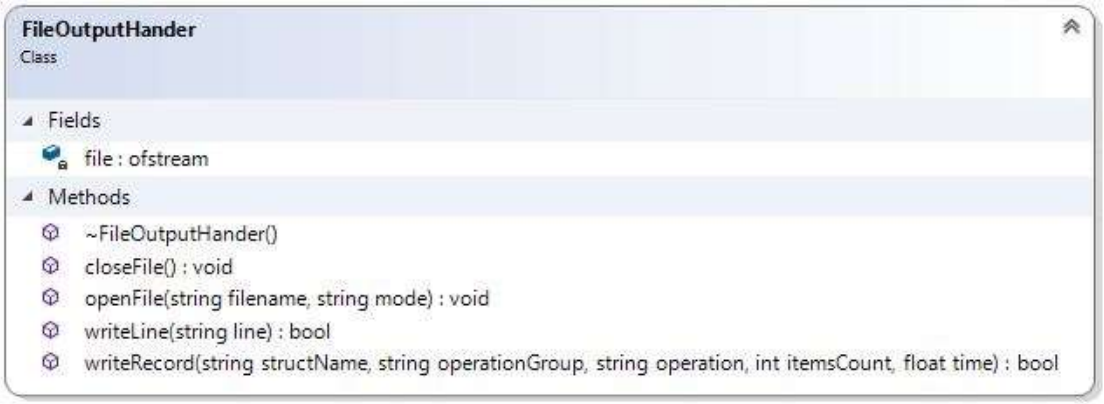
TestInfo

Trieda drží informácie o vykonanom teste. Jednotlivé atribúty sú nastavované pri vykonávaní testu. Obsahuje getre a setre pre atribúty `message` a `operationTime`.



FileOutputHander

Trieda sa stará o vytvorenie a ukladanie CSV súboru pre výstupné dáta testovania. Tiež bude má za úlohu vytvorenie nového záznamu (riadku) v súbore a taktiež aj následné uzatvorenie súboru a jeho uloženie do príslušného priečinka. Pomocou metód pre písanie do súboru sa dá vpísať ľubovoľný jeden riadok alebo riadok vo vopred určenom formáte.



void FileOutputHander::openFile(std::string filename, std::string mode)

Metóda otvorí súbor s daným názvom. Parameter `mode` udáva v akom móde môže byť súbor otvorený `"app"` pre zápis na koniec súboru alebo `"trunc"` pre prepísanie súboru.

void FileOutputHander::closeFile()

Metóda uzavrie súbor pokiaľ bol otvorený.

bool FileOutputHander::writeRecord(std::string structName, std::string operationGroup, std::string operation, int itemsCount, float time)

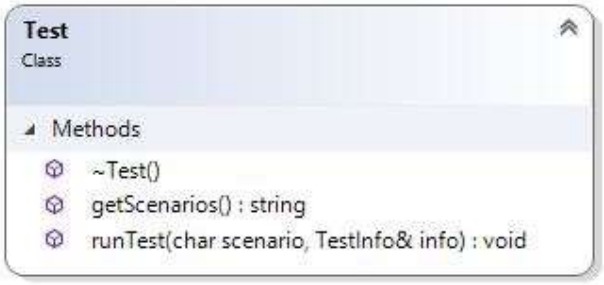
Metóda pre zápis naformátovaného záznamu do otvoreného súboru. Pokiaľ súbor nie je otvorený tak vráti `false`. V opačnom prípade parametre uloží ako nový riadok do súboru, kde tieto údaje oddelí pomocou bodkočiarky a vráti `true`.

bool FileOutputHander::writeLine(std::string line)

Metóda zapíše parameter `line` ako nový riadok do súboru. Pokiaľ súbor nie je otvorený alebo parameter `line` je prázdny string, tak vráti `false`. V opačnom prípade parameter uloží ako nový riadok do súboru a vráti `true`.

Test

Trieda predstavuje rozhranie pre testovacie triedy s implementáciou testu danej štruktúry. Obsahuje virtuálne metódy pre vrátenie množiny testovacích scenárov a tiež spustenie testovacieho scenáru.

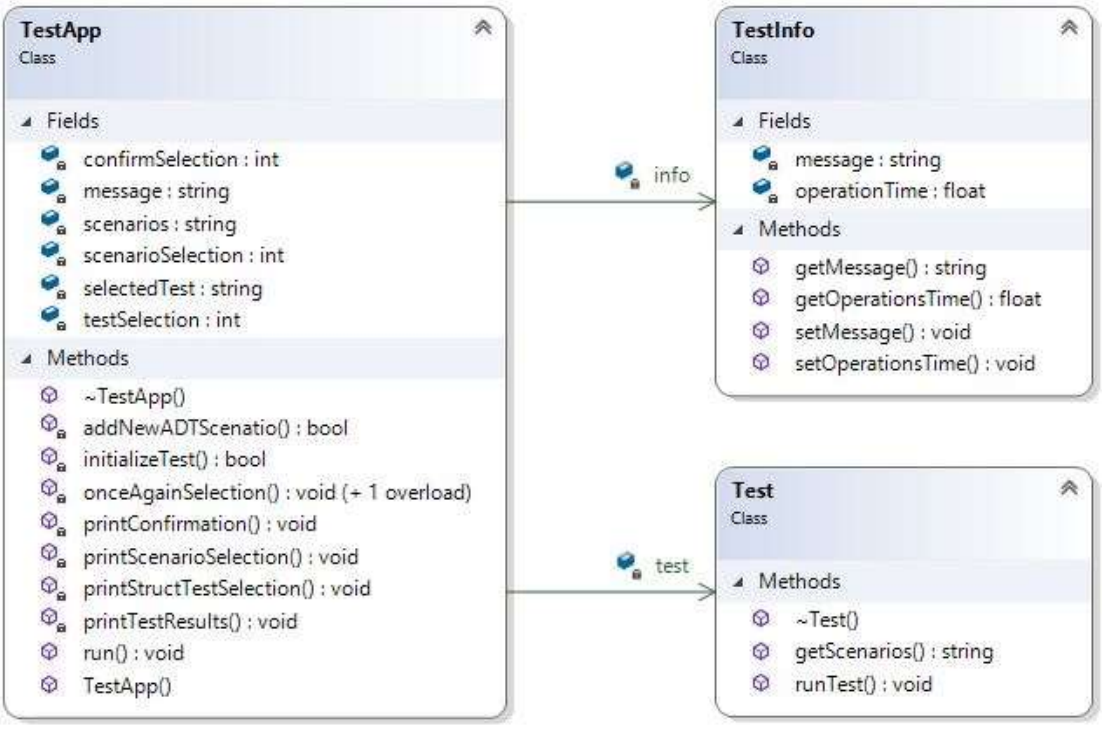


virtual std::string getScenarios()

Virtuálna metóda pre vrátenie množiny testovacích prípadov prislúchajúcich k danej štruktúre. Množina testovacích scenárov je vo formáte `string`.

virtual void runTest(char scenario, TestInfo& info)

Virtuálna metóda pre spustenie scenáru na základe parametra `senario`. Parameter `info` je vkladaný pomocou odkazu a nastavujú sa do neho informácie o prebehnutom teste.



Testovanie ADT zoznamov

Pre test tejto skupiny štruktúr slúži trieda `ADTListTest`. Dedí od triedy `Test` a obsahuje svoje vlastné pomocné metódy pre vykonanie jednotlivých testovacích scenárov. Testujú sa 3 implementácie ADT zoznamu: implementácia poľom, zreťazenou pamťou a obojstranne zreťazeným cyklickým zoznamom. Tieto 3 implementácie si používateľ nemôže navoliť sám, ale sú pevne dané. V rámci jedného testovacieho scenáru sa otestujú tieto implementácie zaradom a výsledok sa zapíše do jedného CSV súboru.

Postup testovania

Používateľ si po spustení aplikácie vyberie v konzole možnosť testovania ADT zoznamov zadaním príslušného vstupu. Trieda `TestApp` vytvorí a nastaví do svojho atribútu typu `Test` novú inštanciu triedy `ADTListTest`. Následne vypíše zoznam možností pre zvolenie testovacieho prípadu do konzoly. Používateľ si po výzve zvolí testovací scenár a trieda `TestApp` spustí test pomocou funkcie triedy `Test` na spustenie testu, kde ako parameter vloží zvolený scenár. O spustenie a správnu implementáciu testovacieho prípadu sa postará trieda `ADTListTest`. Pri vykonávaní testovacieho scenára sa náhodné volanie operácií zabezpečuje generovaním náhodného čísla z intervalu od 1 po 100. Generovanie náhodného čísla sa vykonáva pomocou funkcie `rand()`. Po vygenerovaní tohto čísla sa podľa pomerov jednotlivých skupín operácií pre daný testovací scenár rozhodne, z ktorej skupiny bude vykonaná operácia (Vlož, zruš, sprístupni/nastav, index) a následne po vygenerovaní ďalšieho náhodného čísla pre danú skupinu bude vybraná konkrétna operácia na vykonanie. Výber konkrétnej operácie sa deje s rovnakou pravdepodobnosťou pre skupinu operácií. Pokiaľ nie je možné vykonať operáciu (zavolá sa operácia pre sprístupnenie nad prázdnym zoznamom), bude táto operácia preskočená. Teda výsledný počet operácií v niektorých prípadoch nemusí byť presne 100 000. Každé trvanie vykonanej operácie je zaznamenané v mikrosekundách a následne zapísané ako nový záznam do CSV súboru.

Formát údajov v CSV súbore

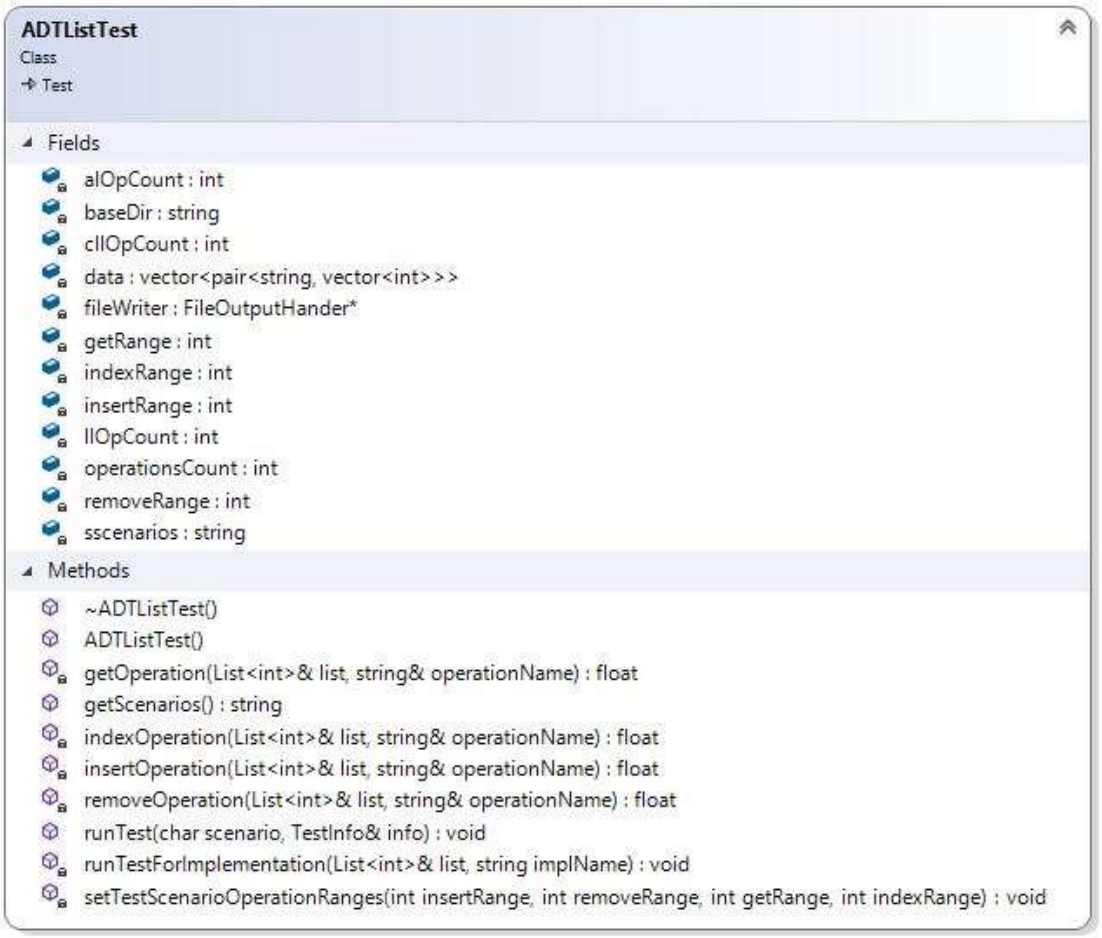
Údaje v CSV súbore zaznamenávajú informácie o jednom celom testovacom scenári pre všetky (3) implementácie údajovej štruktúry v danej úrovni. Jeden záznam (riadok) uchováva informáciu o názve implementácie dátovej štruktúry, skupine vykonanej operácie (napr. vlož, zruš, sprístupni/nastav, index), názvu operácie aká bola vykonaná (vlož na index, zruš prvý, index prvku...), počet prvkov, ktoré bol v štruktúre uložené (pri vkladaní je počet prvkov rovný veľkosti zoznamu pred vložením a pri zrušení zas počtu prvkov pred zrušením) a na koniec je zapísaná časová hodnota doby vykonania danej operácie. Názov testu a testovacieho scenáru je zaznamenaný v názove súboru.

Spracovanie a vyhodnotenie dát

Po vykonaní testu pre jeden testovací scenár bude vygenerovaný príslušný CSV súbor v priečinku `CSV_DATA`. Tento súbor sa importuje do excelu kde sa údaje rozdelené pomocou bodkočiarky v riadkoch rozdelila do jednotlivých buniek tabuľky dát. Surové dáta obsahujú údaje o názve 3 implementácií, názve vykonanej operácie, počte prvkov v zozname a dĺžke trvania operácie v mikrosekundách. Tieto dáta sú v rozdelené v prvom rade bez ohľadu na počet prvkov v zozname na priemerné dĺžky trvania jednotlivých operácií pre jednotlivé implementácie ADT zoznamu. Výsledné priemerné časové údaje sú následne zobrazené v čiarovom diagrame a to v zoskupení podľa implementácie a v zoskupení podľa operácie. Ďalej sa zo surových dát vytvorí kontingenčná tabuľka, kde ako riadok sa vloží počet prvkov v zozname v danom momente a ako hodnoty sa použijú priemery trvania operácie pri danom počte prvkov. To všetko sa zobrazí do stĺpcov rozdelených podľa názvu implementácie ADT zoznamu a ako filter sa použije názov vykonanej operácie.

ADTListTest

Trieda implementuje metódy triedy `Test`, konkrétne pre testovanie ADT zoznamov. Obsahuje aj niektoré vlastné súkromné pomocné metódy. Taktiež trieda obsahuje aj definované vlastné typy ako `clck`, `tp`, `ms`, `duration`. Ide o zjednodušenie použitia C++ tried. Definovaný typ `clck` je pretypovaná trieda `std::chrono::high_resolution_clock`, `tp` reprezentuje triedu `std::chrono::high_resolution_clock::time_point`, `ms` reprezentuje typ časovej jednotky `std::chrono::microseconds` a `duration` reprezentuje triedu pre časový `interval` `std::chrono::duration<float>`. Zoznam testovacích scenárov spolu s ich nastaveniami pomerov výberu operácií je uložený v atribúte `data`, ktorý je typu `std::vector<std::pair<std::string, std::vector<int>>>`



void ADTListTest::runTest(char scenario, TestInfo& info)

Metóda zabezpečuje správne nastavenie pomerov operácií pre zadaný scenár a tiež spustenie vybraného testovacieho scenára. Do parametra `info` sa zapíšu výsledné informácie o prebehnutom teste. V metóde prebehne aj samotné otvorenie súbore pre zápis výsledkov testu v móde prepísania existujúceho súboru.

string ADTListTest::getScenarios()

Metóda vráti množinu scenárov pre ADT zoznamy a to vo forme `string`. Táto množina sa pri zavolaní metódy vždy načíta priamo z CSV súboru určeného pre ukladanie nastavení scenárov pre ADT zoznamy. Scenáre spolu s ich hodnotami pre pomery operácií sa uložia aj do atribútu `data`.

void ADTListTest::setTestScenarioOperationRanges(int insertRange, int removeRange, int getRange, int indexRange)

Seter pre nastavenie pomerov jednotlivých skupín operácií pre zvolený testovací scenár.

void ADTListTest::runTestForImplementation(structures::List<int>& list, string implName)

Pomocná metóda spustí scenár nad implementáciu ADT zoznamu daného ako parameter pomocou odkazu. Parameter `implName` slúži ako názov, ktorý sa má uložiť do CSV súboru pre daná implementáciu.

float ADTListTest::insertOperation(structures::List<int>& list, std::string& operationName)

Pomocná metóda pre vykonanie skupiny operácií pre vloženie prvku. Na začiatku sa vygeneruje náhodné číslo z intervalu od 0 po 2 vrátane a potom sa rozhodne ktorá operácia vloženia sa vykoná. Pokiaľ sa operácia nemôže vykonať tak metóda vráti hodnotu -1. V opačnom prípade ak operácia prebehne metóda vráti jej časovú dobu trvania v mikrosekundách.

float ADTListTest::removeOperation(structures::List<int>& list, std::string& operationName)

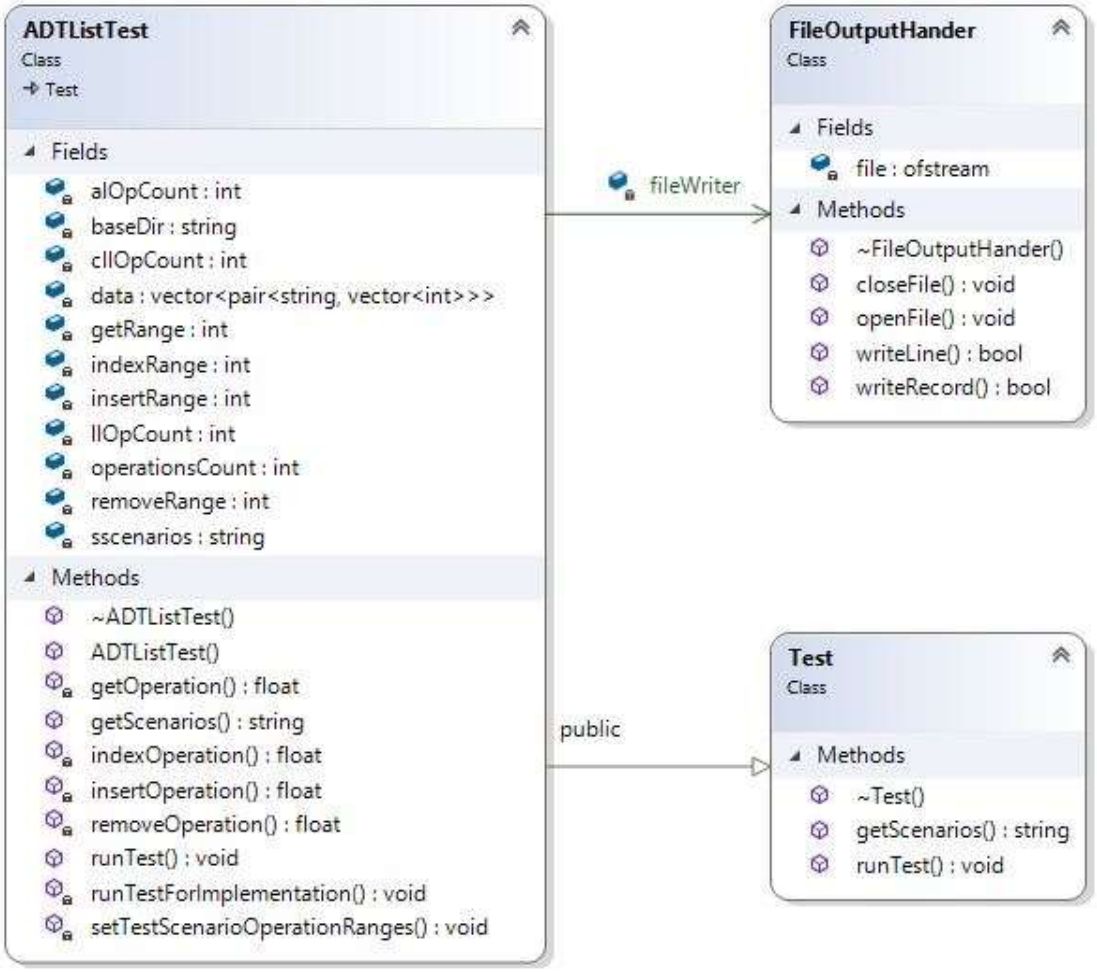
Pomocná metóda pre vykonanie skupiny operácií pre zrušenie prvku. Logika operácie je rovnaká ako pri metóde `insertOperation`. Vracia -1 pokiaľ operácia nebola vykonaná alebo časovú dobu trvania v mikrosekundách pre vykonanú operáciu.

float ADTListTest::getOperation(structures::List<int>& list, std::string& operationName)

Pomocná metóda pre vykonanie skupiny operácií pre sprístupnenie alebo nastavenie prvku. Logika operácie je rovnaká ako pri metóde `insertOperation`. Vracia -1 pokiaľ operácia nebola vykonaná alebo časovú dobu trvania v mikrosekundách pre vykonanú operáciu.

float ADTListTest::indexOperation(structures::List<int>& list, std::string& operationName)

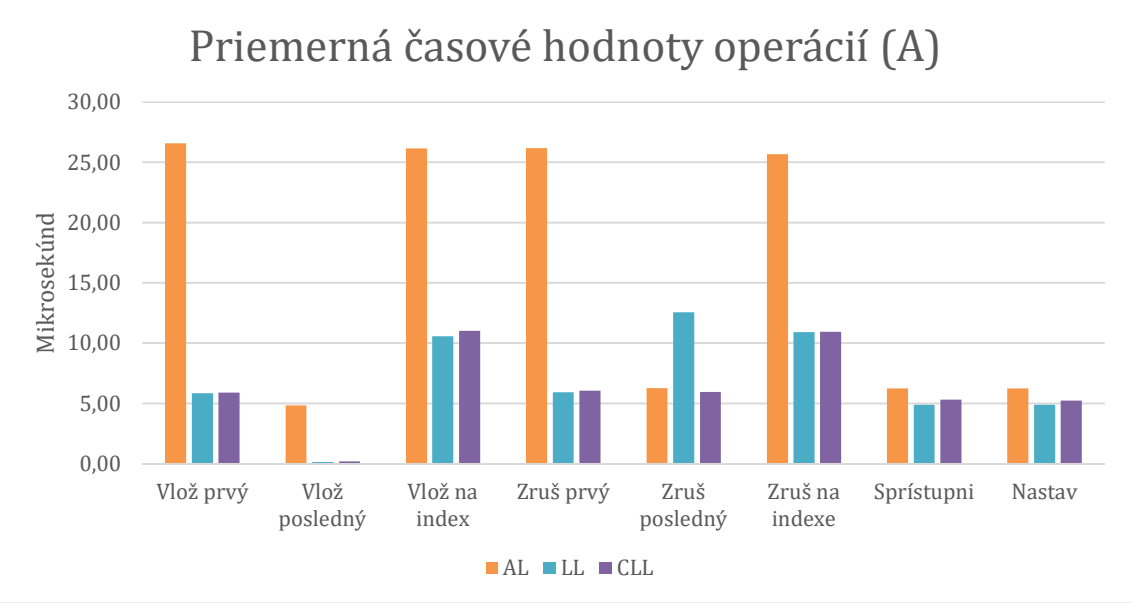
Pomocná metóda pre vykonanie skupiny operácií pre vrátenie indexu prvku. Vracia -1 pokiaľ operácia nebola vykonaná alebo časovú dobu trvania v mikrosekundách pre vykonanú operáciu.

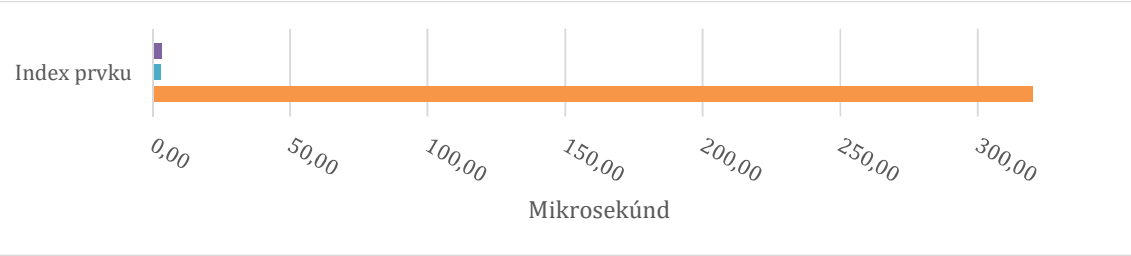

ADTListTest
Class
Test
Fields
alOpCount : int
baseDir : string
clIOpCount : int
data : vector<pair<string, vector<int>>>
getRange : int
indexRange : int
insertRange : int
llOpCount : int
operationsCount : int
removeRange : int
sscenarios : string
Methods
~ADTListTest()
ADTListTest()
getOperation() : float
getScenarios() : string
indexOperation() : float
insertOperation() : float
removeOperation() : float
runTest() : void
runTestForImplementation() : void
setTestScenarioOperationRanges() : void
fileWriter
public
FileOutputHander
Class
Fields
file : ofstream
Methods
~FileOutputHander()
closeFile() : void
openFile() : void
writeLine() : bool
writeRecord() : bool
Test
Class
Methods
~Test()
getScenarios() : string
runTest() : void

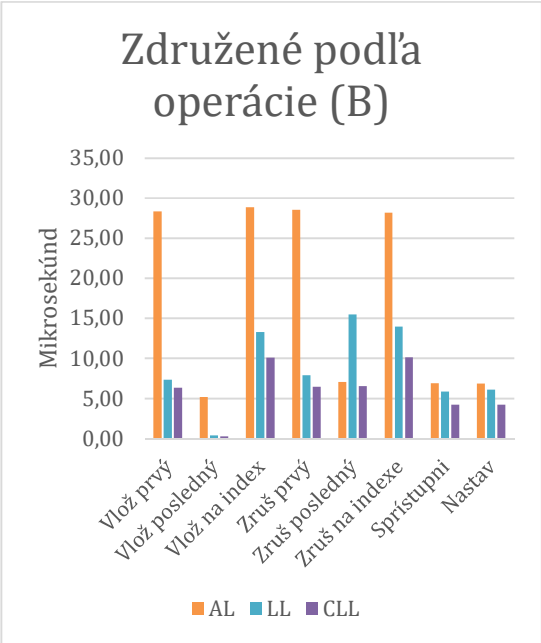
Záver testovania

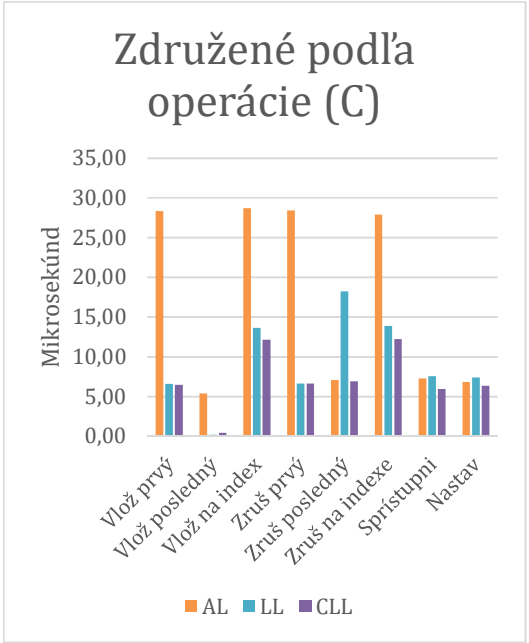
Celková výkonnosť štruktúr

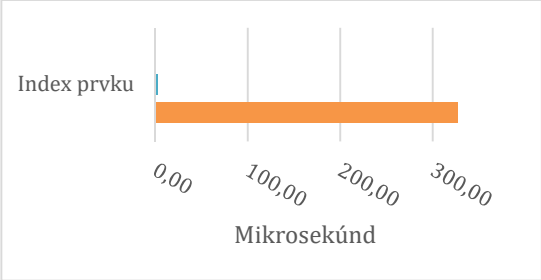
Na priemerných hodnotách časov operácií pre implementácie ADT zoznamu môžeme vidieť, že obojstranne cyklický zoznam sa správa veľmi podobne ako zreťazený zoznam zatiaľ čo pri zozname definovanom poľom sú tieto hodnoty omnoho vyššie a to najmä pri vkladaní alebo rušení prvkov na začiatku zoznamu. Tieto údaje sa v scenároch nemenia ale sú podobné pre každý jeden scenár.

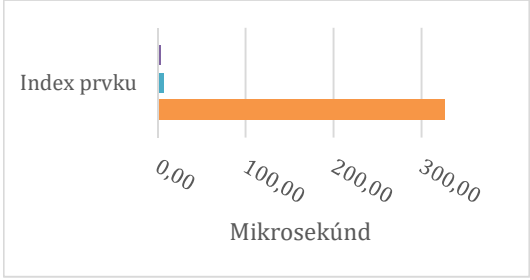






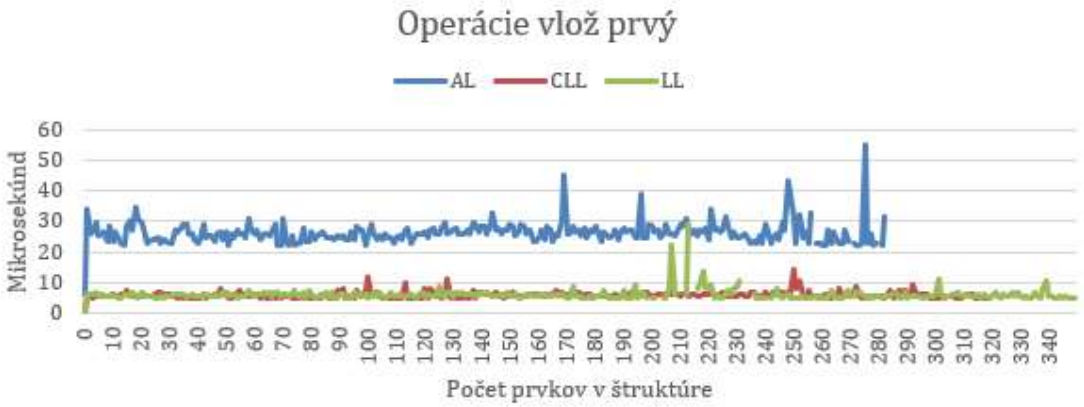




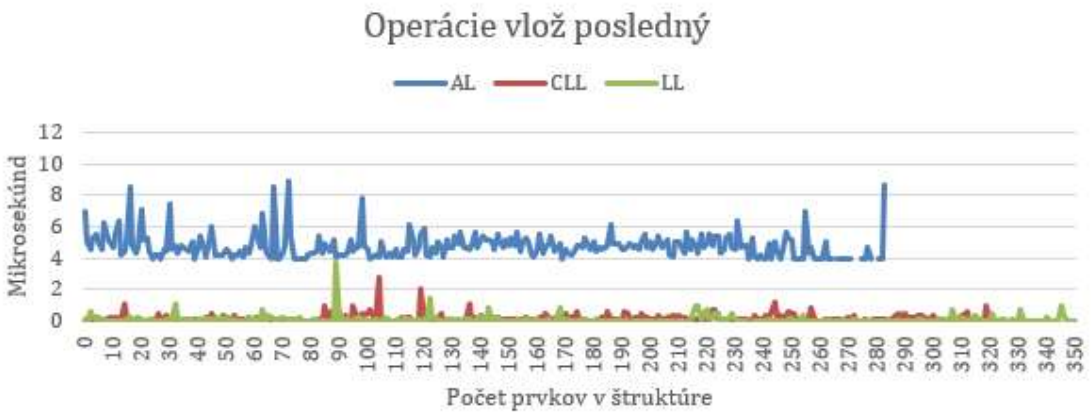


Závislosť jednotlivých operácií od počtu prvkov zoznamu

Časová závislosť operácie od jednotlivého počtu prvkov v zozname je pri každom scenári u toho istého počtu prvkov totožná. V jednotlivých scenároch bolo možné pozorovať len zmenu maximálneho počtu prvkov v zozname(preto som použil na reprezentáciu výsledkov len grafy závislosti času operácie od počtu prvkov len z jedného scenára).



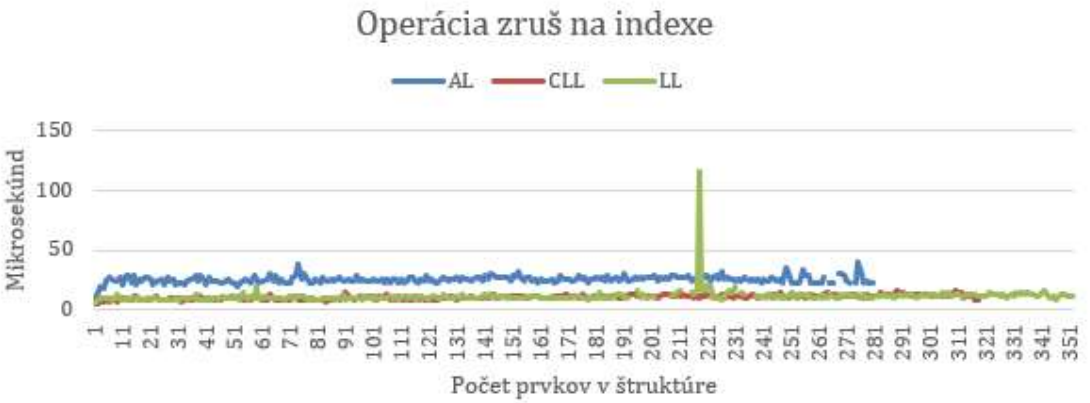
Taktiež pri vkladaní prvkov na začiatok zoznamu je najneefektívnejšia implementácia poľom.



Pri vkladaní môžeme celkovo pozorovať časové výchylky implementácie poľom z dôvodu jeho modifikácie (zväčšenia).

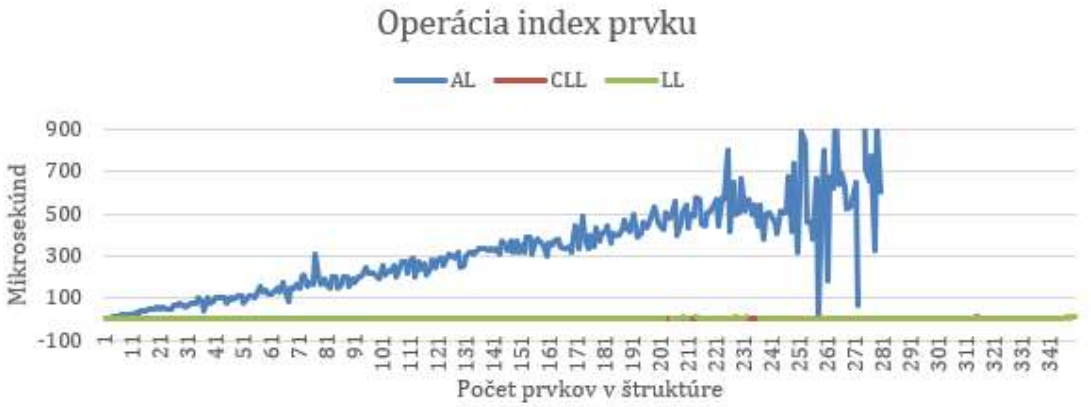
Pri rušení prvkov na začiatku môžeme pozorovať to, že implementácia poľom je veľmi časovo neefektívna už pri malom celkovom počte prvkov v zozname.

Zatiaľ čo pri rušení prvku na konci zoznamu je najneefektívnejšia implementácia zreťazenou pamäťou pričom implementácia poľom je na tom veľmi podobne ako impl. cyklickým zreťazeným zoznamom



Pri operácii zrušenia na indexe môžeme znova pozorovať neefektívnosť implementácie poľom oproti ostatným implementáciám.

Zatiaľ čo pri sprístupnení alebo nastavovaní prvku na index implementácia poľom je ku začiatku trochu neefektívna ale o pár prvkov naviac v zozname môžeme pozorovať naopak jeho efektívnosť oproti ostatným implementáciám.



Testovanie ADT prioritných frontov

Pre test tejto skupiny štruktúr slúži trieda `ADTPriortityQueueTest`. Dedí od triedy `Test` a obsahuje svoje vlastné pomocné metódy pre vykonanie jednotlivých testovacích scenárov. Testujú sa 2 implementácie ADT prioritného frontu: implementácia poľom, ktoré je utriedené podľa priorít ľavostrannou haldou. Tieto 2 implementácie si používateľ nemôže navoliť sám, ale sú pevne dané. V rámci jedného testovacieho scenáru sa otestujú tieto implementácie zaradom a výsledok sa zapíše do jedného CSV súboru

Postup testovania

Používateľ si po spustení aplikácie vyberie v konzole možnosť testovania ADT prioritných frontov zadaním príslušného vstupu. Trieda `TestApp` vytvorí a nastaví do svojho atribútu typu `Test` novú inštanciu triedy `ADTPriortityQueueTest`. Následne vypíše zoznam možností pre zvolenie testovacieho prípadu do konzoly. Používateľ si po výzve zvolí testovací scenár a trieda `TestApp` spustí test pomocou funkcie triedy `Test` na spustenie testu, kde ako parameter vloží zvolený scenár. O spustenie a správnu implementáciu testovacieho prípadu sa postará trieda `ADTPriortityQueueTest`. Pri vykonávaní testovacieho scenára

sa náhodné volanie operácií zabezpečuje generovaním náhodného čísla z intervalu od 1 po 100. Generovanie náhodného čísla sa vykonáva pomocou funkcie `rand()`. Po vygenerovaní tohto čísla sa podľa pomerov jednotlivých skupín operácií pre daný testovací scenár rozhodne, z ktorej skupiny bude vykonaná operácia (Vlož, vyber, a ukáž). Pokiaľ nie je možné vykonať operáciu (zavolá sa operácia pre vybratie prvku nad prázdnym frontom), bude táto operácia preskočená. Teda výsledný počet operácií v niektorých prípadoch nemusí byť presne 100 000. Každé trvanie vykonanej operácie je zaznamenané v mikrosekundách a následne zapísané ako nový záznam do CSV súboru.

Formát údajov v CSV súbore

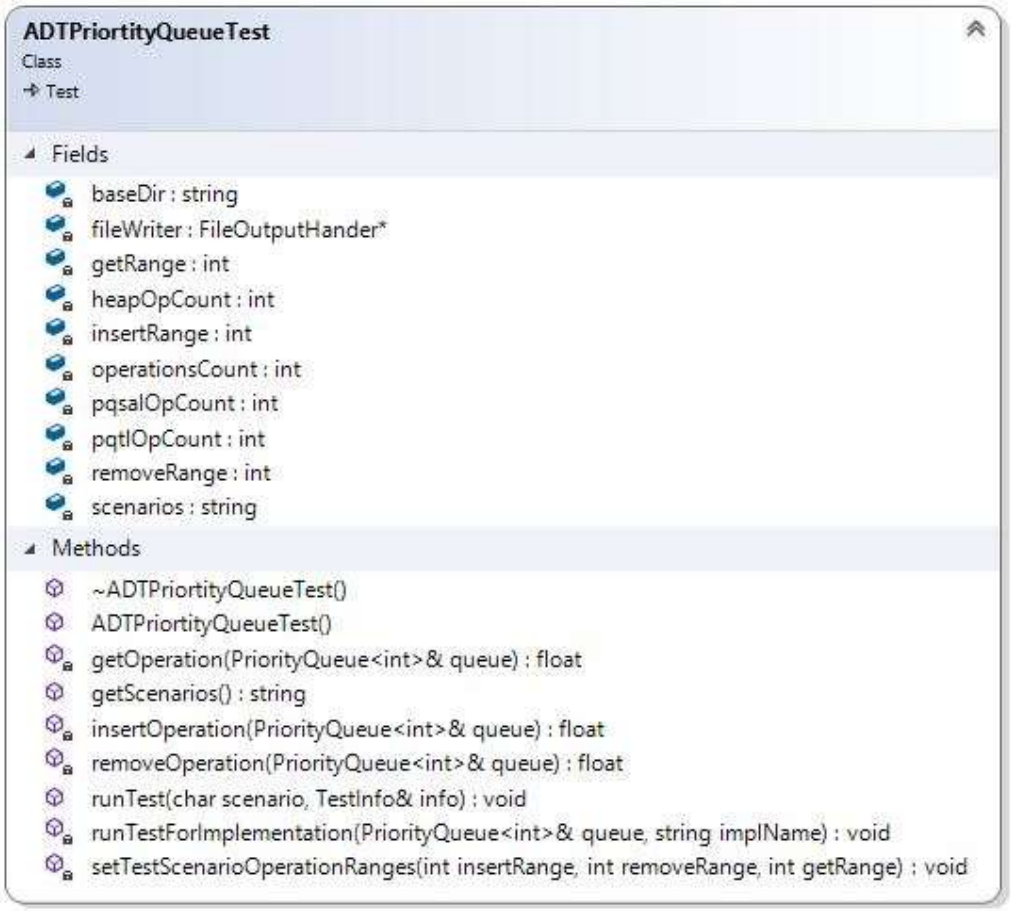
Údaje v CSV súbore zaznamenávajú informácie o jednom celom testovacom scenári pre všetky (2) implementácie údajovej štruktúry v danej úrovni. Jeden záznam (riadok) uchováva informáciu o názve implementácie dátovej štruktúry, názvu operácie aká bola vykonaná(Vlož, vyber, a ukáž), počet prvkov, ktoré bol v štruktúre uložené (pri vkladaní je počet prvkov rovný veľkosti zoznamu pred vložením a pri zrušení zas počtu prvkov pred zrušením) a na koniec je zapísaná časová hodnota doby vykonania danej operácie. Názov testu a testovacieho scenáru je zaznamenaný v názove súboru.

Spracovanie a vyhodnotenie dát

Po vykonaní testu pre jeden testovací scenár bude vygenerovaný príslušný CSV súbor v priečinku `CSV_DATA`. Tento súbor sa importuje do excelu kde sa údaje rozdelené pomocou bodkočiarky v riadkoch rozdelila do jednotlivých buniek tabuľky dát. Surové dáta obsahujú údaje o názve 2 implementácií, názve vykonanej operácie, počte prvkov v zozname a dĺžke trvania operácie v mikrosekundách. Tieto dáta sú v rozdelené v prvom rade bez ohľadu na počet prvkov v zozname na priemerné dĺžky trvania jednotlivých operácií pre jednotlivé implementácie ADT frontu. Výsledné priemerné časové údaje sú následne zobrazené v čiarovom diagrame a to v zoskupení podľa implementácie a v zoskupení podľa operácie. Ďalej sa zo surových dát vytvorí kontingenčná tabuľka, kde ako riadok sa vloží počet prvkov v zozname v danom momente a ako hodnoty sa použijú priemery trvania operácie pri danom počte prvkov. To všetko sa zobrazí do stĺpcov rozdelených podľa názvu implementácie ADT frontu a ako filter sa použije názov vykonanej operácie.

ADTPriortityQueueTest

Trieda implementuje metódy triedy `Test`, konkrétne pre testovanie ADT prioritných frontov. Obsahuje aj niektoré vlastné súkromné pomocné metódy. Taktiež trieda obsahuje aj definované vlastné typy ako `clck`, `tp`, `ms`, `duration`. Ide o zjednodušenie použitia C++ tried. Definovaný typ `clck` je pretypovaná trieda `std::chrono::high_resolution_clock`, `tp` reprezentuje triedu `std::chrono::high_resolution_clock::time_point`, `ms` reprezentuje typ časovej jednotky `std::chrono::microseconds` a `duration` reprezentuje triedu pre časový `interval std::chrono::duration<float>`. Zoznam testovacích scenárov je napevno daný v atribúte typu string.



void ADTPriortityQueueTest::runTest(char scenario, TestInfo& info)

Metóda zabezpečuje správne nastavenie pomerov operácií pre zadaný scenár a tiež spustenie vybraného testovacieho scenára. Do parametra `info` sa zapíšu výsledné informácie o prebehnutom teste. V metóde prebehne aj samotné otvorenie súbore pre zápis výsledkov testu v móde prepísania existujúceho súboru.

std::string ADTPriortityQueueTest::getScenarios()

Metóda vráti množinu scenárov pre ADT zoznamy a to vo forme `string`.

void ADTPriortityQueueTest::setTestScenarioOperationRanges(int insertRange, int removeRange, int getRange)

Seter pre nastavenie pomerov jednotlivých skupín operácií pre zvolený testovací scenár.

void ADTPriortityQueueTest::runTestForImplementation (structures::PriorityQueue<int>& queue, std::string implName)

Pomocná metóda spustí scenár nad implementáciu ADT frontu daného ako parameter pomocou odkazu. Parameter `implName` slúži ako názov, ktorý sa má uložiť do CSV súboru pre danú implementáciu.

float ADTPriortityQueueTest::insertOperation(structures::PriorityQueue<int>& queue)

Pomocná metóda pre vykonanie insert operácie. Vráti časovú hodnotu trvania operácie v mikrosekundách.

float ADTPriortityQueueTest::removeOperation (structures::PriorityQueue<int>& queue)

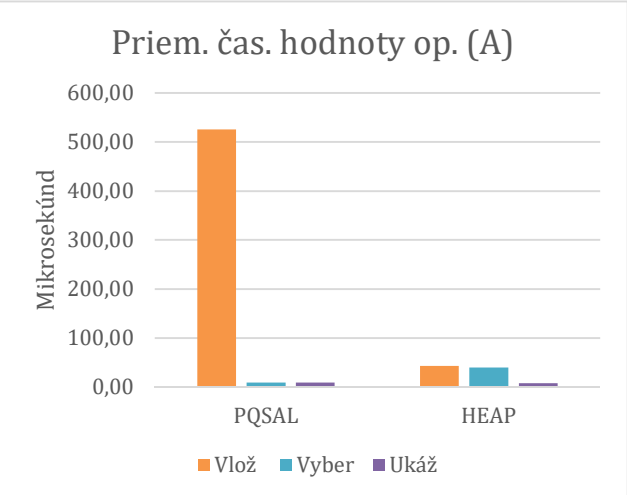
Pomocná metóda pre vykonanie operácie vybratia prvku. Vráti časovú hodnotu trvania operácie v mikrosekundách pokiaľ vo fronte nejaký prvok je inak vráti -1.

float ADTPriortityQueueTest::getOperation(structures::PriorityQueue<int>& queue)

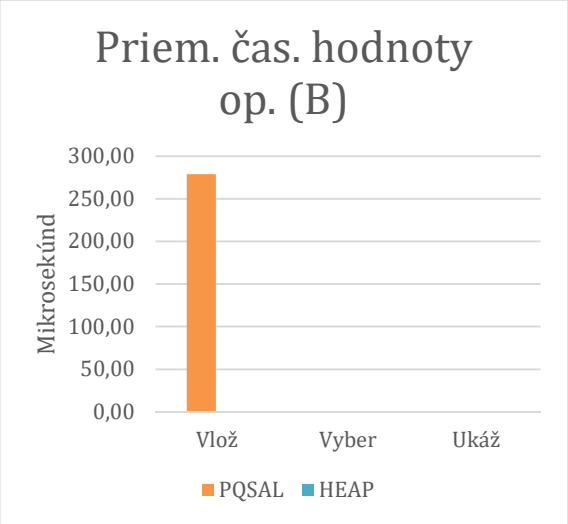
Pomocná metóda pre vykonanie operácie ukázania prvku. Vráti časovú hodnotu trvania operácie v mikrosekundách pokiaľ vo fronte nejaký prvok je inak vráti -1.

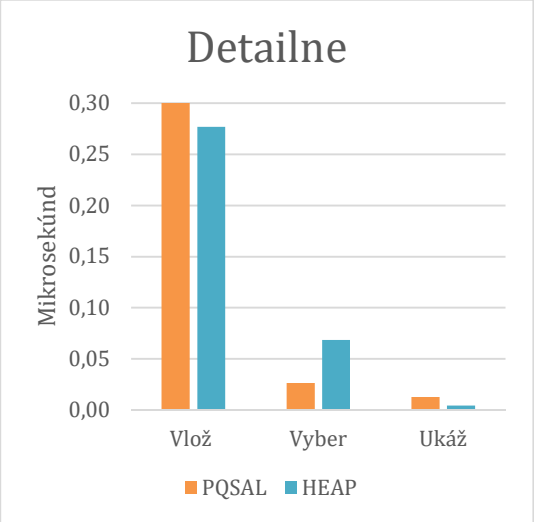
Záver testovania

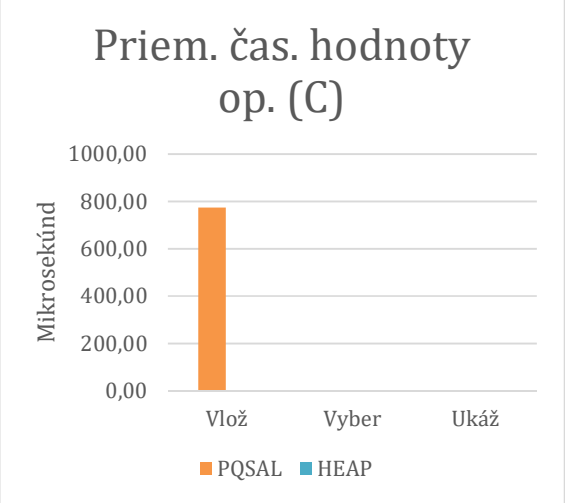
Celková výkonnosť štruktúr

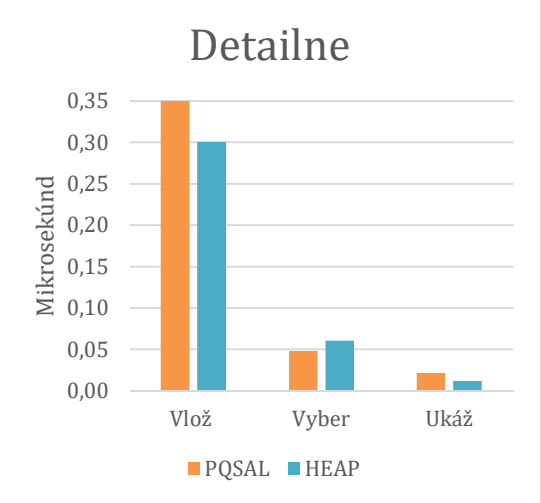


Pri priemerných hodnotách časov opercií pre implementácie ADT prioriného frontu pre jednotlivé scenáre a celková výkonnosť štruktúr nejak výrazne nemení, ale sú tu prítomné menšie zmeny časov. Zatiaľ čo pri implemntácií poľom, ktoré je utriedené podľa priorít môžeme sledovať veľmi rovnakú výkonnosť pri operáciách vo všetkých scenároch tak pri implementácii ľavostrannou haldou vidíme, že priemerný čas operácie je pri scenári A vyšší ako pri ostatných scenároch







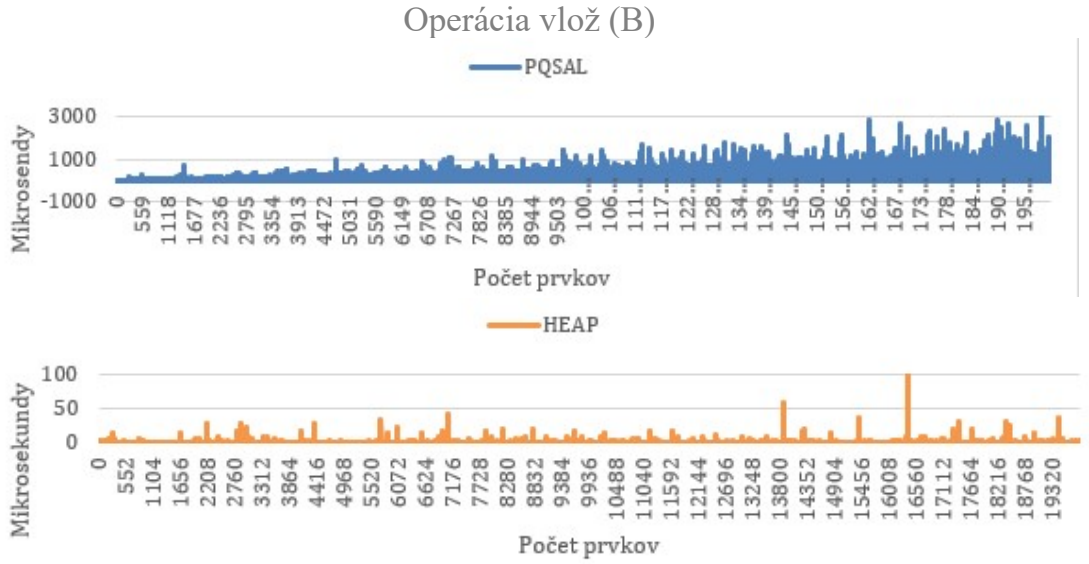


Závislosť jednotlivých operácií od počtu prvkov prioritného frontu

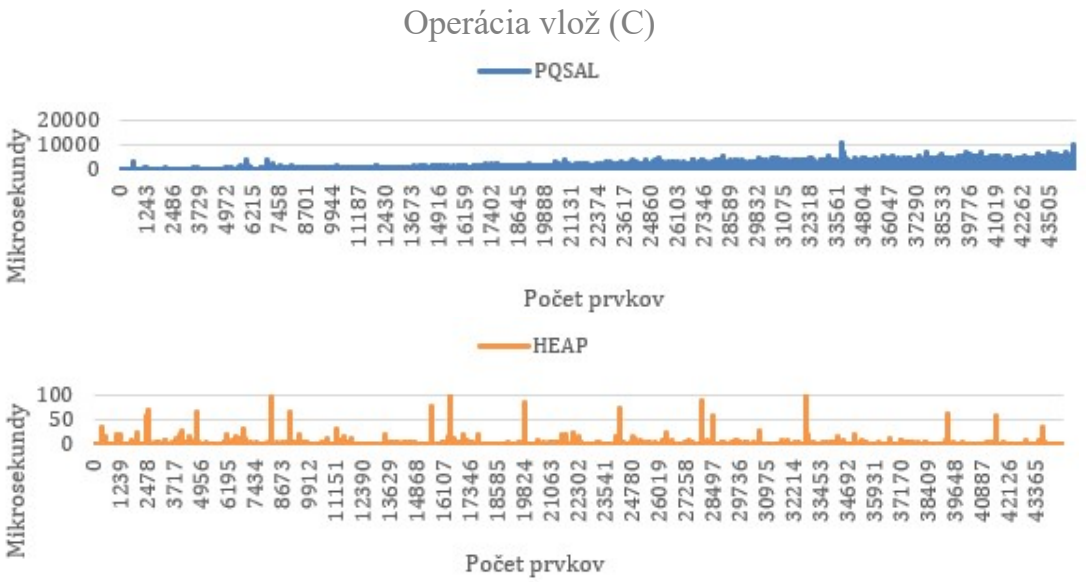
Scenár A:

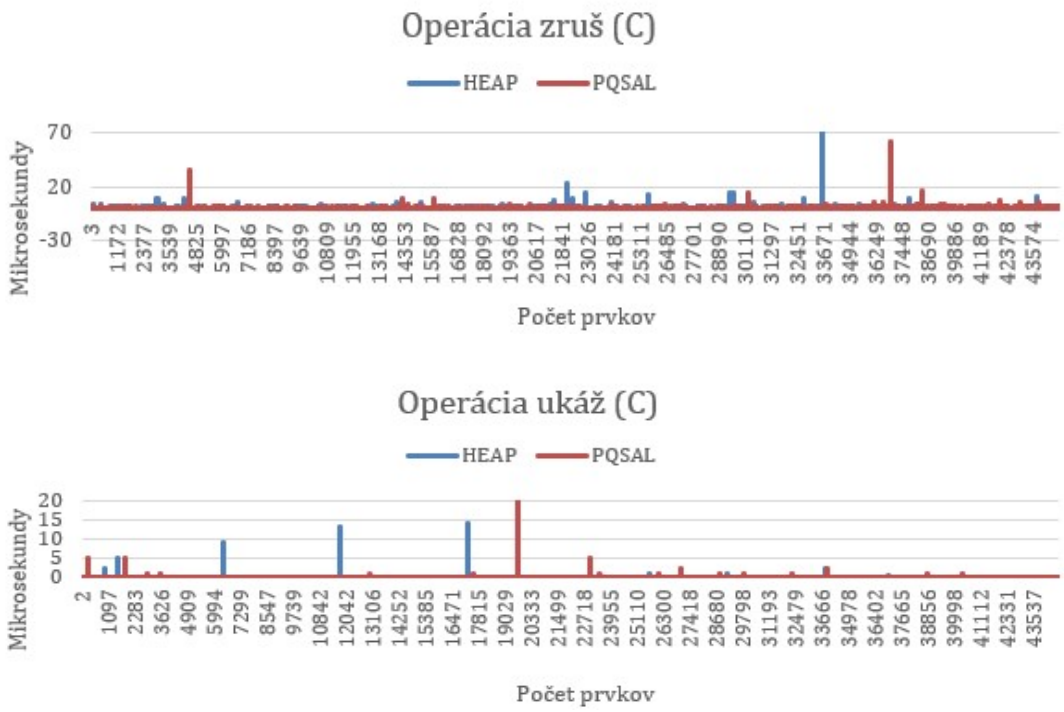
Pri tomto scenári sú závislosti časov operácií podobné obom implementáciám aj keď pri operácii vlož je implementácia utriedeným poľom mierne menej efektívna.

Scenár B a C:
Pri scenári B a taktiež C môžeme pozorovať výrazný rozdiel v efektivite pri operácii vlož. Kde implementácia utriedeným poľom je omnoho neefektívna ako implementácia haldou.



Zatiaľ čo pri operáciách zruš prvok a ukáž prvok sú časy skoro nulové v mikrosekundách





Testovanie ADT viacrozmerných polí

Pre test tejto skupiny štruktúr slúži trieda `MatrixTest`. Dedí od triedy `Test` a obsahuje svoje vlastné pomocné metódy pre vykonanie jednotlivých testovacích scenárov. Testujú sa 2 implementácie ADT viacrozmerného poľa: implementácia v súvislej pamäti a v nesúvislej pamäti. Tieto 2 implementácie si používateľ nemôže navoliť sám, ale sú pevne dané. V rámci jedného testovacieho scenáru sa otestujú tieto implementácie zaradom a výsledok sa zapíše do jedného CSV súboru

Postup testovania

Používateľ si po spustení aplikácie vyberie v konzole možnosť testovania ADT viacrozmerných polí zadaním príslušného vstupu. Trieda `TestApp` vytvorí a nastaví do svojho atribútu typu `Test` novú inštanciu triedy `MatrixTest`. Následne vypíše zoznam možností pre zvolenie testovacieho prípadu do konzoly. Používateľ si po výzve zvolí testovací scenár a trieda `TestApp` spustí test pomocou funkcie triedy `Test` na spustenie testu, kde ako parameter vloží zvolený scenár. O spustenie a správnu implementáciu testovacieho prípadu sa postará trieda `MatrixTest`. Závislosti od parametrov M a N matice sa zisťujú tak, že sa pri maticiach zväčšuje najskôr rozmer M a N je konštantné a následne je to naopak, M je konštantné a N sa zväčšuje.

Formát údajov v CSV súbore

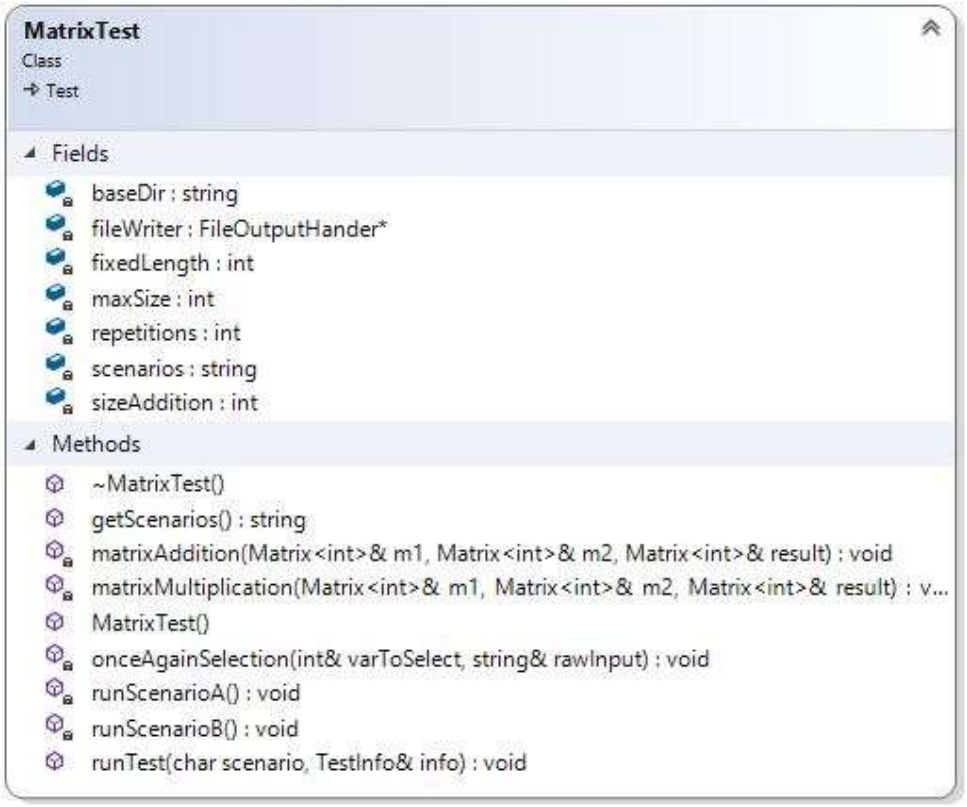
Údaje v CSV súbore zaznamenávajú informácie o jednom celom testovacom scenári pre všetky (2) implementácie údajovej štruktúry v danej úrovni. Jeden záznam (riadok) uchováva informáciu o názve implementácie dátovej štruktúry, názvu operácie aká bola vykonaná(Vlož, vyber, a ukáž), počet prvkov, ktoré bol v štruktúre uložené (pri vkladaní je počet prvkov rovný veľkosti zoznamu pred vložením a pri zrušení zas počtu prvkov pred zrušením) a na koniec je zapísaná časová hodnota doby vykonania danej operácie. Názov testu a testovacieho scenáru je zaznamenaný v názove súboru.

Spracovanie a vyhodnotenie dát

Po vykonaní testu pre jeden testovací scenár bude vygenerovaný príslušný CSV súbor v priečinku `CSV_DATA`. Tento súbor sa importuje do excelu kde sa údaje rozdelené pomocou bodkočiarky v riadkoch rozdelila do jednotlivých buniek tabuľky dát. Surové dáta obsahujú údaje o názve 2 implementácií, veľkosti matice M, N a dĺžke trvania operácie v mikrosekundách. Z týchto surových dát sa najskôr spraví súčet časov pre jednotlivé implementácie. Následne sa z týchto údajov vytvorí kontingenčná tabuľka, ktorá zachytáva časovú závislosť operácie od veľkosti parametra M (z dát kde M sa zvyšuje od 1 po 2000 a N je fixné) a naopak - časovú závislosť operácie od veľkosti parametra N

ADTPriortityQueueTest

Trieda implementuje metódy triedy `Test`, konkrétne pre testovanie ADT viacrozmerných polí. Obsahuje aj niektoré vlastné súkromné pomocné metódy. Taktiež trieda obsahuje aj definované vlastné typy ako `clck`, `tp`, `ms`, `duration`. Ide o zjednodušenie použitia C++ tried. Definovaný typ `clck` je pretypovaná trieda `std::chrono::high_resolution_clock`, `tp` reprezentuje triedu `std::chrono::high_resolution_clock::time_point`, `ms` reprezentuje typ časovej jednotky `std::chrono::microseconds` a `duration` reprezentuje triedu pre časový `interval std::chrono::duration<float>`. Zoznam testovacích scenárov je napevno daný v atribúte typu string.



void MatrixTest::runTest(char scenario, TestInfo& info)

Metóda zabezpečuje spustenie správneho scenára na základe parametra `scenario`. Do parametra `info` sa zapíšu výsledné informácie o prebehnutom teste. V metóde prebehne aj samotné otvorenie súbore pre zápis výsledkov testu v móde prepísania existujúceho súboru. V metóde je prítomné aj nastavenie parametrov pre fixnú dĺžku matice a veľkosť o ktorú má variabilná veľkosť skákať. Toto nastavenie sa realizuje pomocou vstupu používateľa.

std::string MatrixTest::getScenarios()

Metóda vráti množinu scenárov pre ADT zoznamy a to vo forme `string`.

void MatrixTest::onceAgainSelection(int& varToSelect, std::string& rawInput)

Metóda riadenie opätovného vstupu používateľa. Výber používateľa je riadený pomocou číselnej premennej (používateľ zadáva číslo ako svoj výber).

void MatrixTest::runScenarioA()

Metóda pre vykonanie scenáru A, sčitovania 2 matíc. Sčitovanie prebieha v cykloch kde sa menia jednotlivé veľkosti matíc. Pred operáciu sú vygenerované 3 matice s potrebnou veľkosťou. Prvé 2 matice sú inicializované na hodnoty 1 a 2. Tretia je vynulovaná a slúži pre vpísanie výsledkov operácie sčítania. Takéto sčítanie pre daný rozmer matíc sa vykoná 10 krát a výsledný čas sa spriemeruje a následne zapíše do CSV súboru

void MatrixTest::runScenarioB()

Metóda pre vykonanie scenáru B, násobenia 2 matíc. Násobenie prebieha v cykloch kde sa menia jednotlivé veľkosti matíc. Pred operáciu sú vygenerované 3 matice s potrebnou veľkosťou. Prvé 2 matice sú inicializované na náhodné hodnoty. Tretia je vynulovaná a slúži pre vpísanie výsledkov operácie násobenia. Takéto násobenie pre daný rozmer matíc sa vykoná 10 krát a výsledný čas sa spriemeruje a následne zapíše do CSV súboru

void MatrixTest::matrixAddition(mystruct::Matrix<int>& m1, mystruct::Matrix<int>& m2, mystruct::Matrix<int>& result)

Metóda sčíta maticu `m1` s maticou `m2` a výsledok zapíšte do matice `result`.

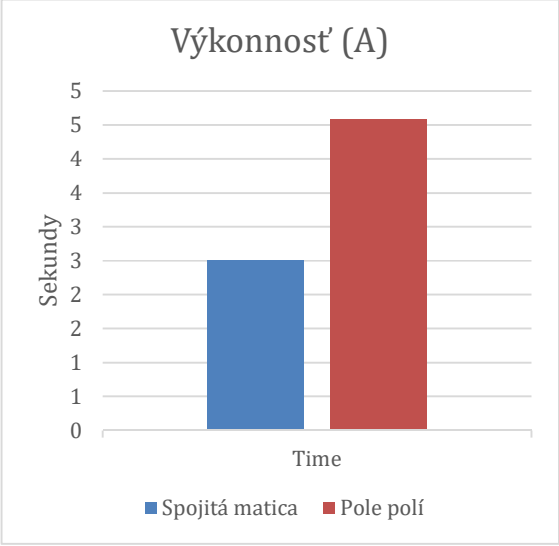
void MatrixTest::matrixMultiplication(mystruct::Matrix<int>& m1, mystruct::Matrix<int>& m2, mystruct::Matrix<int>& result)

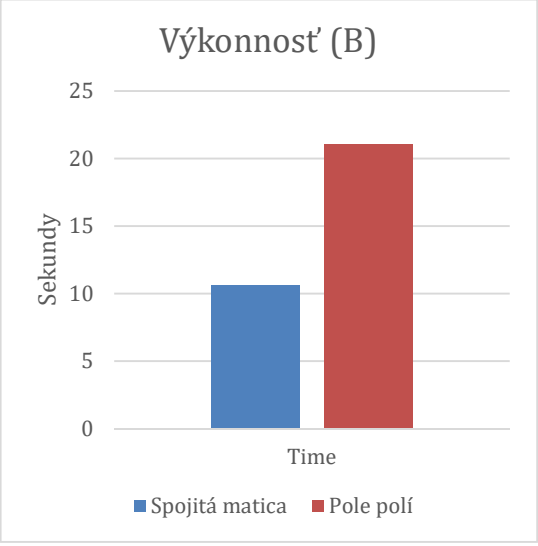
Metóda vynásobí maticu `m1` s maticou `m2` a výsledok zapíšte do matice `result`.

Záver testovania

Celková výkonnosť štruktúr

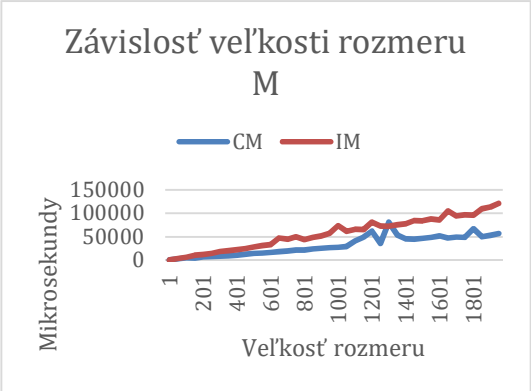
V celkovej výkonnosti štruktúry je najlepšia implementácia spojitou pamäťou oproti implementácii poľom polí a to v oboch scenároch.

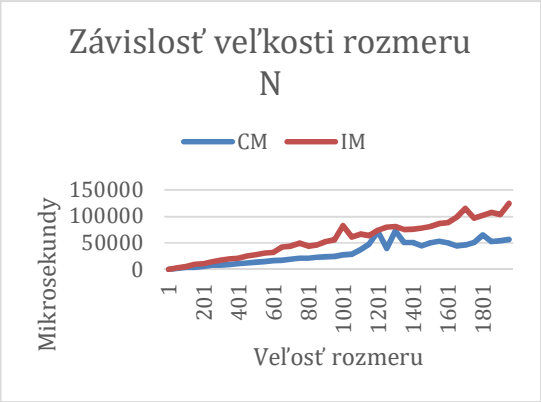




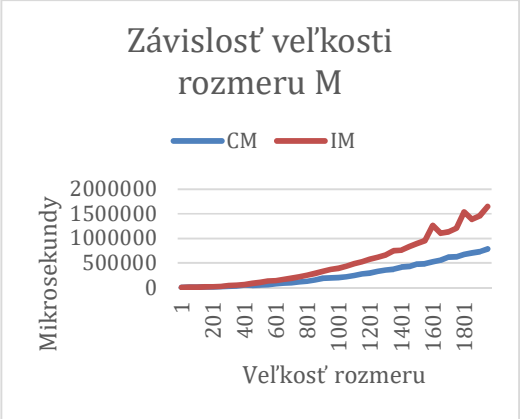
Závislosť parametrov M a N rozmerov matice

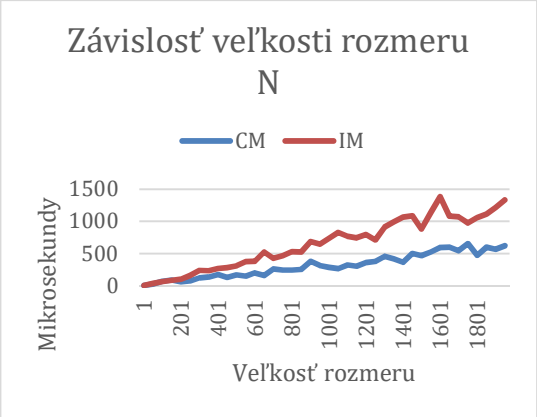
Scenár A:
Pri tomto scenári nie je možné pozorovať rozdiel v časovej závislosti pri zmene veľkosti matice M a N.





Scenár B:
Pri tomto scenári naopak je možné pozorovať rozdiel v časovej závislosti pri zmene veľkosti matice M a N. Parameter M má väčší vplyv na časovú zložitosť operácie. Teda pri zväčšovaní riadku matice je doba operácie väčšia.



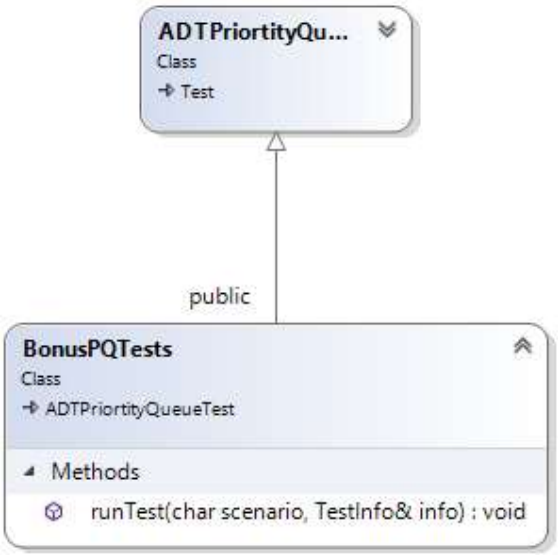


Odhad hornej asymptotickej zložitosti

Pri sčitovaní matíc je horná zložitosť operácie **O(N^2)**, pretože je potrebné prejsť každý prvok matice NxN práve raz. Tým pádom v kóde musím matice prejsť dvoma vnorenými cyklami.
Pri násobení matíc je zložitosť operácie **O(N^3)**. Pri násobení je potrebné prejsť každý prvok násobených matíc aby bolo možné vypočítať všetky prvky výslednej matice. V kóde pri tejto operácii je potrebné matice prejsť tromi vnorenými cyklami.

Testovanie implementácií dvojzoznamu

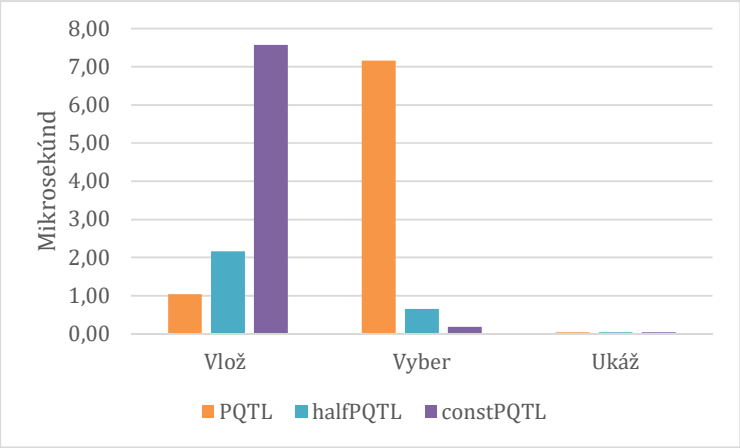
Testovanie implementácií dvojzoznamu ako prioritného frontu je totožné ako testovanie implemtácií prioritných frontov. Z toho dôvodu je vytvorená jedna trieda `BonusPqTests`, ktorá dedí od tredy `ADTPriortityQueueTest` a má len pozmenenú implementáciu metódy pre spustenie test `runTest()`. V tejto metóde sa vykonáva to isté ako pri testovaní prioritných frontov, ale s pozmenenými implementáciami prioritného frontu. A to sú: implementácie dvojzoznamu ako prioritného frontu kde pri prvom je kapacita definovaná konštantne ako počet operácií pre vloženia, v druhej je to odmocnina z celkového počtu prvkov vo fronte a v poslednej implementácii to je dané ako polovica prvkov.



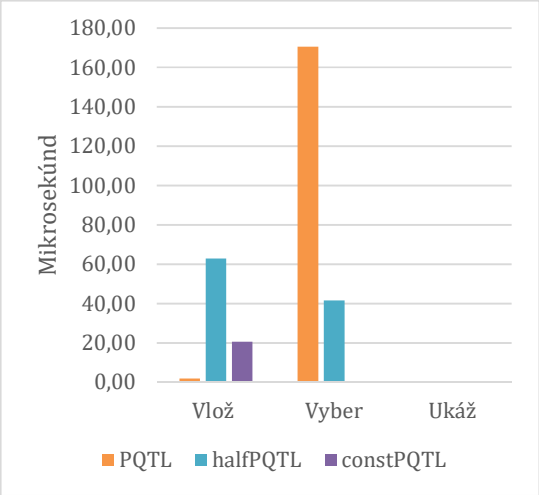
Záver testovania

Celková výkonnosť štruktúr

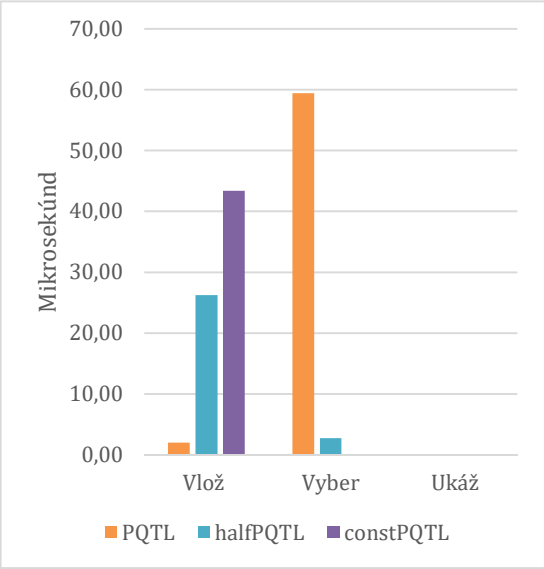
PQTL – dvojzoznam s kapacitou odmocniny počtu pvkov, halfPQTL – kapacita polovice prvkov, constPQTL – konštantná kapacita



Scenár A:
Pri tomto scenári sú rýchlosti operácií vlož a vyber pre konštantnú kapacitu a kapacitu odmocniny priam rovnaké zatiaľ čo operáciu vlož najefektívnejšie zvláda impl. s kapacitou odmocniny. Operáciu Vyber práce naopak zvláda najlepšie impl. s kapacitou konštanty. Výkonnosť operácie ukáž je pri všetkých scenároch totožná.



Scenár B:
Tak isto ako pri scenári A tak aj pri B je pri operácii vloženia najefektívnejšia impl. s kapacitou odmocniny. Pri operácii ukázania prvku je to impl. s konštantnou kapacitou. V tomto scenári môžene pozorovať nárast času vykonania oprácie výberu pri impl. s kapacitou odmocnina



Scenár C:
Tak isto ako pri scenári A a B tak aj pri C je pri operácii vloženia najefektívnejšia impl. s kapacitou odmocniny. Pri operácii ukázania prvku je to impl. s konštantnou kapacitou. Môžeme ale pozorovať nárast priemerného času operácie vlož pre impl. s konštantnou kapacitou.