Report za letný semester

---

1. Implementácia nových metód a transformácií:

1.1 Hromadné operácie (Overloading *push_back_all*)

Pre zjednodušenie importu dát z rôznych zdrojov boli naimplementované štyri varianty metódy *push_back_all*:

1. **Z iného ResizableArray:** Sekvenčne pridá všetky prvky z kompatibilnej inštancie.
2. **Zo surového C-pola s explicitnou veľkosťou *(const T* arr, size_t size)*:** Bezpečne načíta dáta z dynamických polí.
3. **Zo statického C-pola *(template<size_t N> void push_back_all(const T (&arr)[N]))*:** Šablónový variant, kde veľkosť poľa odvodí kompilátor automaticky počas kompilácie.
4. **Zo štandardného kontajnera *std::vector*:** Umožňuje priamu kompatibilitu s STL knižnicou.

1.2 Výbery, transformácie a iterátory

- *sub_rarray(size_t from, size_t to)*: Extrahovanie explicitného rozsahu indexov do novej inštancie poľa s kontrolou validity indexov (*std::out_of_range*).
- *filter(Predicate pred)*: Šablónová metóda pracujúca s predikátom (lambda výrazom). Vytvorí nové pole obsahujúce iba prvky vyhovujúce zadaného podmienke.
- *flatten()*: Transformačná operácia pre hierarchické vrstvenie (pole polí), ktorá „sploští“ viacrozmernú štruktúru do jedného lineárneho ResizableArray.
- **Podpora iterátorov (Iterator a ConstIterator):** Implementácia vlastných tried iterátorov s preťažením operátorov (operator*, operator++, operator!=) a metód *begin()* / *end()*. Týmto bola sprístupnená podpora pre moderné cykly založené na rozsahu (**range-based for loops**).

1.3 Špecializovaný in-place Merge Sort (sort())

Navrhnutý a naimplementovaný bol **Merge Sort** striktne upravený pre prácu na mieste (**in-place**) s pomocnou pamäťovou náročnosťou iba *O(1)* pre dátový typ T. Algoritmus najprv lokálne zoradí samostatné fyzické bloky (DataBlock) pomocou *std::sort*, následne premietne ich hranice na virtuálnu os indexov a hierarchicky ich zlučuje pomocou rotácie prvkov a presúvacej sémantiky (*std::move*), čím rešpektuje špecifickú blokovú architektúru kontajnera.

---

2. Dokumentácia zdrojového kódu

Súčasťou vývoja v letnom semestri bolo vypracovanie komplexnej technickej dokumentácie priamo v zdrojovom kóde. Všetky metódy boli detailne zdokumentované podľa priemyselného štandardu **Doxygen**. Každá funkcia obsahuje jasný opis svojho účelu, definíciu očakávaných vstupných parametrov (*@param*), špecifikáciu návratových hodnôt (*@return*), zoznam vyvolávaných výnimiek (*@throw*) a typové parametre šablón (*@tparam*). Dokumentácia bola navrhnutá tak, aby ju bolo možné automaticky vygenerovať do prehľadného HTML alebo PDF formátu, čo výrazne zjednodušuje údržbu kódu, zvyšuje jeho čitateľnosť pre ostatných vývojárov a zabezpečuje splnenie formálnych akademických kritérií kladených na softvérové projekty.

---

3. Jednotkové testy (Unit Testing)

V nadväznosti na prísne testovanie zimy bol rozšírený testovací modul v rámci **Google Test** (*test_basic.cpp*). Boli úspešne overené všetky hraničné a operačné stavy:

- **Transformácie a rozhranie:** Overenie správnosti pridávania dát z rôznych typov polí cez *push_back_all*, presnosť filtrácie pomocou lambdy a správne správanie iterátorov v cykloch.
- **Hraničné stavy triedenia (SortTest):** Scenáre SmallArray (zoradenie v rámci jedného bloku), EmptyAndSingleElement (stabilita pri poliach s veľkosťou 0 alebo 1), AlreadySorted (správanie pri vopred zoradenom poli) a DuplicateValues (stabilita pri duplicitách).
- **Záťažové stavy:** Scenár MultipleLevelsMerge potvrdil funkčnosť zlučovania na hraniciach blokov medzi úplne odlišnými úrovňami štruktúry a scenár RandomLargeArray úspešne otestoval triedenie masívneho počtu náhodných hodnôt.

Všetky unit testy boli úspešne vykonané a skončili s návratovým kódom exit code 0, čo potvrdzuje stopercentnú validitu správy pamäte a správnosť algoritmov.

---

4. Výsledok a záver

Výsledkom práce v letnom semestri je plne dokončená, produkčne stabilná a optimalizovaná dátová štruktúra, ktorá úspešne spája výhody hierarchického ukladania dát s flexibilitou štandardných STL kontajnerov. Implementáciou iterátorov a hromadných operácií sa výrazne zjednodušilo a zmodernizovalo klientske programové rozhranie.

Kľúčovým prínosom tejto fázy projektu je úspešný návrh algoritmu sort(), ktorý dokázal vyriešiť netriviálnu úlohu in-place triedenia v nepravidelne rozptýlenej pamäti bez

potreby dodatočnej alokácie RAM. Robustnosť celého riešenia, jeho pamäťová bezpečnosť a správnosť algoritmov boli spoľahlivo overené sadou jednotkových testov, čím bol projekt úspešne dovedený do finálneho a uceleného stavu.