My Project

Generated by Doxygen 1.13.2

Chapter 1

Release aprašymai

1.1 v.pradinė

Įvadimai studentų duomenys ir ir išvedami jų apskaičiuoti vidurkiai ar medianos į komdų eilutę

1.2 v0.1

Galima generuoti studentų duomenis ir sukurtos 2 projekto versijos 1 - naudojant std::vector, 2 - naudojant c masyvus

1.3 V0.2

Pridėtas duomenų įvesties į ir išvesties iš failo pasirinkimas, panaikinta c masyvą naudojanti programos versija

1.4 V0.3

Pridėtas duomenų rūšiavimas pagal pasirinktus kriterijus

1.5 V0.3.1

Klaidų žinutės lietuvių kalba

1.6 V0.4

Sukurtas README.md failas kuriami surašyti programos spartos matmenys ir pridėtas studentų failo generavimo funkcionalumas

1.7 V0.4.1

Pataisyti ir papildyti matavimai, klaidų ištaisymas

1.8 V0.4.2

Klaidos pataisymas

1.9 V1.0 pradinis relizas

Sukurtos papildomai dvi programos veikimo versijos, papildytas README programos veikimo spartos matmenimis ir testavimo detalėmis

1.10 V1.1

Studentas perkeltas į klasę

1.11 V1.2

Relizuota "Rule of five" ir įvesties/išvesties operatorius Student klasėje

1.12 V1.5

Sukurta bazinė klasė žmogus(Person) ir pritaikyta Studnet klasei

1.13 V2.0

Sukurti Doxygen HTML ir PDF dokumentacija, pridėti unit testai naudojant gtest

1.14 Naudojimos instrukcija

- Paleidimas: Nueiti į norimo naudojimo konteinerio build aplankalą ir paleisti programą "1-oji-uzduotis"
- Skaičiavimų atlikimas:
    1. Paleidus programą duodamas pasirinkimas 1-6, jeigu norite atlikti skaičiavimus su studentai pasirinkite 1-4 pagal poreikį (ką pasirinkimai daro bus aprašyta programos veikimo metu)
    2. Jeigu pasirinkote 4 pasirinkimą įvesikte studentų failo pavadinimą (studentų failas turi būti tame pačiame aplankale iš kurios ir paleidote programą)
    3. Pasirinkite rūšiavimo būdą (vidurkių ir medianų rūšiavimas yra nuo mažiausio iki didžiausio)
    4. Pasirinkite išvesties būdą
    5. Pasirinkite skirstymo strategiją
    6. Jeigu pasirinkote įrašyti pažimius ranka ar generuoti juos sekite pragramoje nurodytas instrukcijas

1.15 Diegimo instrukcija

1. Nueikite į norimos programos versijos aplankalo (Deque/List/Vector)
   - `cd Vector`
     Arba
   - `cd List`
     Arba
   - `cd Deque`
2. Sukurkite build aplankalą ir įeikite į jį
   `mkdir build && cd build`
3. Paleiskite cmake comandą
   `cmake ..`
4. Išeikite iš aplankalo ir sukompiliuokite programą
   `cd .. && cmake --build build`
5. Pragrama bus build aplankale su pavadinimu "1-oji-uzduotis"

1.16 Tyrimai

1.16.0.1 Matavimo sistemos parametrai:

CPU: 7800x3d \ **RAM**: 32GB | 6000 MT/s \ **SSD**: NVMe M.2 | 7,000/7,000MB/s read/write

1.16.1 Pirmas tyrimas (Failų kūrimas ir jų uždarymas)

- Kiekvienas studentas turi po 10 namų darbų rezultatų + egzaminas
- Laiko matavimui naudoajam `chrono` biblioteka
- Matavimai atliekami penkis kartus ir iš jų išgaunami vidurkiai
- Kodas buvo sukompiliuotas naudojant -O3 gaire

Studentu sk.	Laikas 1	Laikas 2	Laikas 3	Laikas 4	Laikas 5	Vidurkiai
1,000	866 s	966 s	877 s	792 s	936 s	887 s
10,000	6486 s	6903 s	6497 s	6364 s	6307 s	6511 s
100,000	68 ms	66 ms	65 ms	66 ms	67 ms	66 ms
1,000,000	594 ms	592 ms	587 ms	595 ms	608 ms	595 ms
10,000,000	5917 ms	5827 ms	5871 ms	5843 ms	5887 ms	5869 ms

>Failo kūrimo pavizdys:

1.16.2 Antrasis tyrimas (Duomenų apdorojimas)

- Laiko matavimui naudoajam `chrono` biblioteka
- Duomenys gaunaumi iš ankščiau sugeneruotų failų
- Kiekvienas studentas turi po 10 namų darbų rezultatų + egzaminas
- Matavimai atliekami penkis kartus ir iš jų išgaunami vidurkiai
- Matuojami vykdomi su penkiais skirtingais failo dydžiais
- Kodas buvo sukompiliuotas naudojant -O3 gaire
- Matuojami:
  - nuskaitymas iš failo
  - studentų rūšiavimą į dvi grupes/kategorijas
  - surūšiuotų studentų išvedimą į du naujus failus
  - visos programos veikimo laikas

1.16.2.1 1,000 studentų

Matavimas	Laikas 1	Laikas 2	Laikas 3	Laikas 4	Laikas 5	Vidurkiai
Nuskaitymas	545 s	639 s	518 s	509 s	485 s	539 s
Rūšiavimas	101 s	134 s	100 s	101 s	101 s	107 s
Išvedimas	340 s	435 s	394 s	362 s	369 s	380 s
Veikimo laikas	1092 s	1334 s	1113 s	1076 s	1056 s	1134 s

1.16.2.2 10,000 studentų

Matavimas	Laikas 1	Laikas 2	Laikas 3	Laikas 4	Laikas 5	Vidurkiai
Nuskaitymas	5524 s	5118 s	5430 s	5809 s	5024 s	5381 s
Rūšiavimas	1043 s	1129 s	1133 s	1107 s	1106 s	1104 s
Išvedimas	2795 s	2835 s	2870 s	2859 s	2770 s	2826 s
Veikimo laikas	10321 s	10053 s	10386 s	10760 s	9873 s	10279 s

1.16.2.3 100,000 studentų

Matavimas	Laikas 1	Laikas 2	Laikas 3	Laikas 4	Laikas 5	Vidurkiai
Nuskaitymas	47 ms	48 ms	46 ms	49 ms	47 ms	47 ms
Rūšiavimas	7 ms	7 ms	6 ms	6 ms	6 ms	6 ms
Išvedimas	27 ms	27 ms	25 ms	32 ms	33 ms	29 ms
Veikimo laikas	90 ms	92 ms	88 ms	98 ms	97 ms	93 ms

1.16.2.4 1,000,000 studentų

Matavimas	Laikas 1	Laikas 2	Laikas 3	Laikas 4	Laikas 5	Vidurkiai
Nuskaitymas	440 ms	443 ms	443 ms	444 ms	444 ms	443 ms
Rūšiavimas	51 ms	51 ms	53 ms	50 ms	52 ms	51 ms
Išvedimas	270 ms	273 ms	275 ms	267 ms	268 ms	270 ms
Veikimo laikas	848 ms	855 ms	859 ms	849 ms	853 ms	853 ms

1.16.2.5 10,000,000 studentų

Matavimas	Laikas 1	Laikas 2	Laikas 3	Laikas 4	Laikas 5	Vidurkiai
Nuskaitymas	4476 ms	4483 ms	4502 ms	4465 ms	4491 ms	4483 ms
Rūšiavimas	668 ms	677 ms	685 ms	670 ms	679 ms	676 ms
Išvedimas	2602 ms	2628 ms	2586 ms	2640 ms	2665 ms	2624 ms
Veikimo laikas	8686 ms	8750 ms	8730 ms	8727 ms	8781 ms	8735 ms

1.16.3 Trečias tyrimas (Konteinerių testavimas)

- Laiko matavimui naudoajam `chrono` biblioteka
- Duomenys gaunaumi iš ankščiau sugeneruotų failų
- Kiekvienas studentas turi po 10 namų darbų rezultatų + egzaminas
- Matavimai atliekami penkis kartus ir iš jų išgaunami vidurkiai
- Matuojami vykdomi su penkiais skirtingais failo dydžiais
- Kodas buvo sukompiliuotas naudojant -O3 gaire
- Duomenys rūšiuojami studento pažymių vidurkio didėjimo tvarka
- Matuojas:
  - Duomenų nuskaitymas iš failų į atitinkamą konteinerį
  - Studentų rūšiavimas didėjimo tvarką konteineryje
  - Studentų skirstymas į dvi grupes/kategorijas
- Studentams saugoti bus naudojami 4 skirtingi konteineriai:
  - `std::vector`
  - `std::list`
  - `std::deque`
  - Mano sukurta `Vector` klasė

1.16.4 Naudojant `std::vector`

1.16.4.1 1,000 studentų

Matavimas	Laikas 1	Laikas 2	Laikas 3	Laikas 4	Laikas 5	Vidurkiai
Nuskaitymas	504 s	573 s	582 s	609 s	495 s	552 s
Rušiavimas	53 s	65 s	67 s	64 s	52 s	60 s
Skirstymas	98 s	119 s	112 s	122 s	99 s	110 s

1.16.4.2 10,000 studentų

Matavimas	Laikas 1	Laikas 2	Laikas 3	Laikas 4	Laikas 5	Vidurkiai
Nuskaitymas	5396 s	5124 s	5232 s	4846 s	5028 s	5125 s
Rušiavimas	826 s	745 s	785 s	939 s	943 s	847 s
Skirstymas	955 s	941 s	965 s	997 s	1099 s	991 s

1.16.4.3 100,000 studentų

Matavimas	Laikas 1	Laikas 2	Laikas 3	Laikas 4	Laikas 5	Vidurkiai
Nuskaitymas	49040 s	46073 s	48257 s	49288 s	50360 s	48603 s
Rušiavimas	11722 s	12060 s	11927 s	12071 s	12136 s	11983 s
Skirstymas	9479 s	9184 s	8919 s	9446 s	9274 s	9260 s

1.16.4.4 1,000,000 studentų

Matavimas	Laikas 1	Laikas 2	Laikas 3	Laikas 4	Laikas 5	Vidurkiai
Nuskaitymas	430 ms	435 ms	437 ms	443 ms	436 ms	436 ms
Rušiavimas	134 ms	140 ms	145 ms	141 ms	136 ms	139 ms
Skirstymas	111 ms	109 ms	110 ms	110 ms	109 ms	109 ms

1.16.4.5 10,000,000 studentų

Matavimas	Laikas 1	Laikas 2	Laikas 3	Laikas 4	Laikas 5	Vidurkiai
Nuskaitymas	4414 ms	4401 ms	4396 ms	4444 ms	4414 ms	4413 ms
Rušiavimas	1743 ms	1665 ms	1729 ms	1722 ms	1660 ms	1703 ms
Skirstymas	1350 ms	1357 ms	1348 ms	1343 ms	1350 ms	1349 ms

1.16.5 Naudojant `std::list`

1.16.5.1 1,000 studentų

Matavimas	Laikas 1	Laikas 2	Laikas 3	Laikas 4	Laikas 5	Vidurkiai
Nuskaitymas	560 s	576 s	606 s	485 s	594 s	564 s
Rušiavimas	52 s	64 s	66 s	52 s	65 s	59 s
Skirstymas	99 s	121 s	119 s	94 s	126 s	111 s

1.16.5.2 10,000 studentų

Matavimas	Laikas 1	Laikas 2	Laikas 3	Laikas 4	Laikas 5	Vidurkiai
Nuskaitymas	4843 s	5330 s	4923 s	4755 s	5391 s	5048 s
Rušiavimas	761 s	738 s	756 s	757 s	735 s	749 s
Skirstymas	1005 s	1068 s	894 s	882 s	963 s	962 s

1.16.5.3 100,000 studentų

Matavimas	Laikas 1	Laikas 2	Laikas 3	Laikas 4	Laikas 5	Vidurkiai
Nuskaitymas	47977 s	47000 s	48757 s	49032 s	59094 s	50372 s
Rušiavimas	12449 s	12087 s	12294 s	12273 s	12074 s	12235 s
Skirstymas	9721 s	9184 s	10342 s	9746 s	9834 s	9765 s

1.16.5.4 1,000,000 studentų

Matavimas	Laikas 1	Laikas 2	Laikas 3	Laikas 4	Laikas 5	Vidurkiai
Nuskaitymas	476 ms	472 ms	477 ms	477 ms	473 ms	475 ms
Rušiavimas	285 ms	233 ms	256 ms	243 ms	233 ms	250 ms
Skirstymas	167 ms	155 ms	168 ms	165 ms	162 ms	163 ms

1.16.5.5 10,000,000 studentų

Matavimas	Laikas 1	Laikas 2	Laikas 3	Laikas 4	Laikas 5	Vidurkiai
Nuskaitymas	4643 ms	4681 ms	4723 ms	4707 ms	4613 ms	4673 ms
Rušiavimas	5225 ms	5597 ms	5502 ms	5185 ms	5349 ms	5371 ms
Skirstymas	1647 ms	1710 ms	1669 ms	1654 ms	1658 ms	1667 ms

1.16.6 Naudojant `std::deque`

1.16.6.1 1,000 studentų

Matavimas	Laikas 1	Laikas 2	Laikas 3	Laikas 4	Laikas 5	Vidurkiai
Nuskaitymas	589 s	511 s	480 s	565 s	593 s	547 s
Rušiavimas	133 s	108 s	100 s	124 s	126 s	118 s
Skirstymas	102 s	84 s	81 s	99 s	91 s	91 s

1.16.6.2 10,000 studentų

Matavimas	Laikas 1	Laikas 2	Laikas 3	Laikas 4	Laikas 5	Vidurkiai
Nuskaitymas	4974 s	4865 s	4524 s	4909 s	4793 s	4813 s
Rušiavimas	1103 s	1213 s	1129 s	1070 s	1106 s	1124 s
Skirstymas	891 s	814 s	760 s	852 s	768 s	817 s

1.16.6.3 100,000 studentų

Matavimas	Laikas 1	Laikas 2	Laikas 3	Laikas 4	Laikas 5	Vidurkiai
Nuskaitymas	45226 s	45315 s	57134 s	46732 s	45353 s	47952 s
Rušiavimas	13310 s	13076 s	13947 s	12625 s	12929 s	13177 s
Skirstymas	8509 s	8437 s	8048 s	8168 s	8224 s	8277 s

1.16.6.4 1,000,000 studentų

Matavimas	Laikas 1	Laikas 2	Laikas 3	Laikas 4	Laikas 5	Vidurkiai
Nuskaitymas	453 ms	461 ms	458 ms	452 ms	453 ms	455 ms
Rušiavimas	159 ms	166 ms	155 ms	164 ms	164 ms	161 ms
Skirstymas	128 ms	132 ms	140 ms	136 ms	127 ms	132 ms

1.16.6.5 10,000,000 studentų

Matavimas	Laikas 1	Laikas 2	Laikas 3	Laikas 4	Laikas 5	Vidurkiai
Nuskaitymas	4520 ms	4489 ms	4540 ms	4530 ms	5122 ms	4640 ms
Rušiavimas	2040 ms	1952 ms	1952 ms	1982 ms	1969 ms	1979 ms
Skirstymas	1496 ms	1479 ms	1503 ms	1482 ms	1500 ms	1492 ms

1.16.7 Naudojant naują Vector klasę

1.16.7.1 1,000 studnetų

Matavimas	Laikas 1	Laikas 2	Laikas 3	Laikas 4	Laikas 5	Vidurkiai
Nuskaitymas	582 s	565 s	576 s	696 s	596 s	603 s
Rušiavimas	133 s	129 s	129 s	129 s	136 s	131 s
Skirstymas	152 s	146 s	151 s	156 s	144 s	150 s

1.16.7.2 10,000 studentų

Matavimas	Laikas 1	Laikas 2	Laikas 3	Laikas 4	Laikas 5	Vidurkiai
Nuskaitymas	6914 s	7813 s	6592 s	6308 s	6574 s	6840 s
Rušiavimas	1801 s	1753 s	1473 s	1493 s	1813 s	1667 s
Skirstymas	2303 s	2246 s	2304 s	2066 s	2123 s	2208 s

1.16.7.3 100,000 studentų

Matavimas	Laikas 1	Laikas 2	Laikas 3	Laikas 4	Laikas 5	Vidurkiai
Nuskaitymas	56841 s	53579 s	54079 s	58699 s	54874 s	55614 s
Rušiavimas	18657 s	19323 s	19014 s	18681 s	18608 s	18857 s
Skirstymas	17205 s	12676 s	12705 s	13748 s	12706 s	13808 s

1.16.7.4 1,000,000 studentų

Matavimas	Laikas 1	Laikas 2	Laikas 3	Laikas 4	Laikas 5	Vidurkiai
Nuskaitymas	524 ms	508 ms	508 ms	507 ms	498 ms	509 ms
Rušiavimas	232 ms	232 ms	227 ms	232 ms	236 ms	232 ms
Skirstymas	291 ms	286 ms	276 ms	257 ms	270 ms	276 ms

1.16.7.5 10,000,000 studentų

Matavimas	Laikas 1	Laikas 2	Laikas 3	Laikas 4	Laikas 5	Vidurkiai
Nuskaitymas	6073 ms	5933 ms	5660 ms	5653 ms	5420 ms	5758 ms
Rušiavimas	2954 ms	2883 ms	2832 ms	2834 ms	2942 ms	2889 ms
Skirstymas	3855 ms	3704 ms	3552 ms	3355 ms	3298 ms	3553 ms

1.16.8 Ketvirtas tyrimas (Skirstymas)

- Laiko matavimui naudoajam `chrono` biblioteka

- Duomenys gaunaumi iš ankščiau sugeneruotų failų
- Kiekvienas studentas turi po 10 namų darbų rezultatų + egzaminas
- Matavimai atliekami penkis kartus ir iš jų išgaunami vidurkiai
- Kodas buvo sukompiliuotas naudojant -O3 gaire
- Matuojami vykdomi su penkiais skirtingais failo dydžiais
- Duomenys nerūšiuojami
- Naudojamos 3 strategijos:

1. Bendro studentai konteinerio skaidymas (rūšiavimas) į du naujus to paties tipo konteinerius
2. Bendro studentų konteinerio skaidymas (rūšiavimas) panaudojant tik vieną naują konteinerį
3. Bendro studentų konteinerio skaidymas (rūšiavimas) panaudojant "efektyvius" darbo su konteineriais metodus (`erase`, `remove_if`)

1.16.8.1 Vector

1.16.8.2 1 Strategija

Studentu sk.	Laikas 1	Laikas 2	Laikas 3	Laikas 4	Laikas 5	Vidurkiai
1,000	86 s	82 s	103 s	87 s	80 s	87 s
10,000	1064 s	1212 s	1122 s	1155 s	1119 s	1134 s
100,000	7906 s	7346 s	7287 s	7355 s	7335 s	7445 s
1,000,000	52 ms	53 ms	53 ms	49 ms	53 ms	52 ms
10,000,000	711 ms	707 ms	701 ms	699 ms	707 ms	705 ms

1.16.8.3 2 Strategija

Studentu sk.	Laikas 1	Laikas 2	Laikas 3	Laikas 4	Laikas 5	Vidurkiai
1,000	58 s	70 s	71 s	60 s	69 s	65 s
10,000	807 s	761 s	796 s	769 s	737 s	774 s
100,000	6122 s	5950 s	6083 s	6338 s	5944 s	6087 s
1,000,000	46 ms	47 ms	46 ms	46 ms	47 ms	46 ms
10,000,000	571 ms	570 ms	560 ms	562 ms	564 ms	565 ms

1.16.8.4 3 Strategija

Studentu sk.	Laikas 1	Laikas 2	Laikas 3	Laikas 4	Laikas 5	Vidurkiai
1,000	36 s	41 s	48 s	43 s	40 s	41 s
10,000	616 s	562 s	571 s	614 s	614 s	595 s
100,000	4613 s	4101 s	4130 s	4014 s	3839 s	4139 s
1,000,000	28 ms	28 ms	29 ms	29 ms	33 ms	29 ms
10,000,000	355 ms	356 ms	354 ms	355 ms	351 ms	354 ms

1.16.8.5 List

1.16.8.6 1 Strategija

Studentu sk.	Laikas 1	Laikas 2	Laikas 3	Laikas 4	Laikas 5	Vidurkiai
1,000	138 s	103 s	97 s	103 s	119 s	112 s
10,000	1207 s	1020 s	1117 s	953 s	994 s	1058 s
100,000	9856 s	9813 s	9280 s	9690 s	10011 s	9730 s
1,000,000	96 ms	100 ms	97 ms	97 ms	98 ms	97 ms
10,000,000	946 ms	956 ms	956 ms	948 ms	1000 ms	961 ms

1.16.8.7 2 Strategija

Studentu sk.	Laikas 1	Laikas 2	Laikas 3	Laikas 4	Laikas 5	Vidurkiai
1,000	17 s	21 s	18 s	25 s	16 s	19 s
10,000	218 s	177 s	175 s	181 s	179 s	186 s
100,000	1699 s	1693 s	1693 s	1682 s	1678 s	1689 s
1,000,000	16 ms	17 ms	17 ms	17 ms	16 ms	16 ms
10,000,000	172 ms	172 ms	173 ms	170 ms	170 ms	171 ms

1.16.8.8 3 Strategija

Studentu sk.	Laikas 1	Laikas 2	Laikas 3	Laikas 4	Laikas 5	Vidurkiai
1,000	60 s	50 s	63 s	49 s	50 s	54 s
10,000	529 s	517 s	674 s	553 s	677 s	590 s
100,000	5377 s	5615 s	5088 s	5006 s	5225 s	5262 s
1,000,000	66 ms	69 ms	68 ms	68 ms	66 ms	67 ms
10,000,000	668 ms	668 ms	685 ms	665 ms	676 ms	672 ms

1.16.8.9 Deque

1.16.8.10 1 Strategija

Studentu sk.	Laikas 1	Laikas 2	Laikas 3	Laikas 4	Laikas 5	Vidurkiai
1,000	84 s	85 s	138 s	107 s	84 s	99 s
10,000	844 s	804 s	786 s	898 s	832 s	832 s
100,000	10858 s	8951 s	8304 s	7947 s	8524 s	8916 s
1,000,000	82 ms	84 ms	80 ms	80 ms	80 ms	81 ms
10,000,000	820 ms	818 ms	810 ms	820 ms	819 ms	817 ms

1.16.8.11 2 Strategija

Studentu sk.	Laikas 1	Laikas 2	Laikas 3	Laikas 4	Laikas 5	Vidurkiai
1,000	37 s	34 s	33 s	35 s	29 s	33 s
10,000	248 s	249 s	250 s	266 s	257 s	254 s
100,000	2483 s	2414 s	2526 s	2483 s	2520 s	2485 s
1,000,000	24 ms	24 ms	24 ms	24 ms	23 ms	23 ms
10,000,000	243 ms	253 ms	239 ms	243 ms	243 ms	244 ms

1.16.8.12 3 Strategija

Studentu sk.	Laikas 1	Laikas 2	Laikas 3	Laikas 4	Laikas 5	Vidurkiai
1,000	47 s	48 s	53 s	57 s	58 s	52 s
10,000	460 s	446 s	452 s	425 s	436 s	443 s
100,000	4231 s	4184 s	4566 s	4238 s	4094 s	4262 s
1,000,000	49 ms	46 ms	48 ms	49 ms	48 ms	48 ms
10,000,000	465 ms	466 ms	467 ms	465 ms	468 ms	466 ms

1.16.9 Penktas tyrimas (Struct vs Class)

- Laiko matavimui naudoajam `chrono` biblioteka
- Duomenys gaunaumi iš ankščiau sugeneruotų failų
- Kiekvienas studentas turi po 10 namų darbų rezultatų + egzaminas
- Matavimai atliekami penkis kartus ir iš jų išgaunami vidurkiai
- Matuojami du kodo atvėjai, Vienas naudojant Struktūras studentams kitas klases
- Matavimui naudajami tik 1,000,000 ir 10,000,000 studentų kiekiai
- Matavimui bus lyginami -O1, -O2 ir -O3 kompiliavmo gairės (3 atvėjai, kodo veikimo greitis ir bin failo dydis)
- Matuojamas pilnas programos veikimo laikas (neskaitant vartotojo įvesties)
- Naudojamas vector tipo konteineris studentų masyui saugoti

1.16.9.1 Veikimo greitis -O1

(stud. sk.) Duom. tipas	Laikas 1	Laikas 2	Laikas 3	Laikas 4	Laikas 5	Vidurkiai
(1,000,000) Klasė	1562 ms	1619 ms	1616 ms	1630 ms	1593 ms	1604 ms
(1,000,000) Struktūra	1306 ms	1333 ms	1321 ms	1328 ms	1311 ms	1319 ms
(10,000,000) Klasė	17244 ms	17597 ms	17800 ms	17520 ms	17590 ms	17550 ms
(10,000,000) Struktūra	13950 ms	14018 ms	14012 ms	14125 ms	14019 ms	14024 ms

Klasės metodo bin dydis: 107.94 KiB Struktūros metodo bin dydis: 110.88 KiB

1.16.9.2 Veikimo greitis -O2

(stud. sk.) Duom. tipas	Laikas 1	Laikas 2	Laikas 3	Laikas 4	Laikas 5	Vidurkiai
(1,000,000) Klasė	1559 ms	1607 ms	1599 ms	1606 ms	1597 ms	1593 ms
(1,000,000) Struktūra	1443 ms	1338 ms	1337 ms	1359 ms	1358 ms	1367 ms
(10,000,000) Klasė	16920 ms	17292 ms	17082 ms	17048 ms	17276 ms	17123 ms
(10,000,000) Struktūra	14104 ms	14132 ms	14284 ms	14277 ms	14272 ms	14213 ms

Klasės metodo bin dydis: 117.92 KiB Struktūros metodo bin dydis: 119.91 KiB

1.16.9.3 Veikimo greitis -O3

(stud. sk.) Duom. tipas	Laikas 1	Laikas 2	Laikas 3	Laikas 4	Laikas 5	Vidurkiai
(1,000,000) Klasė	1569 ms	1556 ms	1550 ms	1551 ms	1572 ms	1559 ms
(1,000,000) Struktūra	1264 ms	1329 ms	1291 ms	1278 ms	1292 ms	1290 ms
(10,000,000) Klasė	16396 ms	16528 ms	16528 ms	16639 ms	16589 ms	16536 ms
(10,000,000) Struktūra	13311 ms	13454 ms	13388 ms	13425 ms	13332 ms	13382 ms

Klasės metodo bin dydis: 129.52 KiB Struktūros metodo bin dydis: 123.69 KiB

Chapter 2

Vectoriaus klasė

2.1 Vectoriaus metodai

2.1.1 push_back()

push_back funkcija įdeda elementą į masyvo galą

```
Vector<int> my_vector;

my_vector.push_back(20);

std::cout « my_vector.at(0) « std::endl;
// Išveda 20
```

2.1.2 pop_back()

pop_back funkcija išėma elementą iš masyvo galo ir gražino jo vertę

```
Vector<int> my_vector;

my_vector.push_back(20);
my_vector.push_back(5);

std::cout « my_vector.pop_back() « std::endl;
// Išveda 20
std::cout « my_vector.pop_back() « std::endl;
// Išveda 5
```

2.1.3 at()

Gražina nuorodą (refrence) į masyvo elementą nurodytame indekse. Jei indeksas už masyvo ribų išmetamas erroras

```
Vector<int> my_vector;

my_vector.push_back(20);

std::cout « my_vector.at(0) « std::endl;
// Išveda 20

std::cout « my_vector.at(1) « std::endl;
// std::out_of_range
// at() index out of range
```

2.1.4 size()

Gražina masyvo elementų kiekį

```
Vector<int> my_vector;

my_vector.push_back(20);
my_vector.push_back(5);
my_vector.push_back(-7);

std::cout « my_vector.size() « std::endl;
// Išveda 3
```

2.1.5 capacity()

Gražina reservuotą masyvo elementų skaičių

```
Vector<int> my_vector;

my_vector.push_back(20);
my_vector.push_back(5);
my_vector.push_back(-7);

std::cout « my_vector.capacity() « std::endl;
// Išveda 4
```

2.2 Spartos analizė

- Laiko matavimui naudoajam `chrono` biblioteka
- Testavimo eiga:
  1. Pradedamas laiko matavimas
  2. Sukuriamas tuščias vector konteineris su int tipo parametru
  3. Vektorius užpildomas didėjančia skaičiū seka iki nustatyto elementų kiekio
  4. Sustabdomas laiko skaičiavimas
  5. Sunaikinamas vector konteineris
- Testai pakartojami po 10 kartų ir iš jū gaunamas vidurkis

Elementų kiekis	std::vector greitis	Vector greitis
10000	7.883e-06s	1.9581e-05s
100000	0.00025s	0.00018s
1000000	0.00178s	0.00123s
10000000	0.00998s	0.01192s
100000000	0.07464s	0.0928s

Chapter 3

Class Index

3.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

Chapter 4

File Index

4.1 File List

Here is a list of all documented files with brief descriptions:

Chapter 5

Class Documentation

5.1 Vector< T > Class Template Reference

Public Member Functions

- **Vector** (const Vector< T > &other)
- Vector< T > & **operator=** (const Vector< T > &other)
- **Vector** (Vector< T > &&other) noexcept
- Vector< T > & **operator=** (Vector< T > &&other) noexcept
- T & **operator[ ]** (const size_t index)
- bool **operator==** (const Vector< T > &other)
- bool **operator!=** (const Vector< T > &other)
- void **assign** (const size_t count, const T &value)
- void **push_back** (const T &value)
- T **pop_back** ()
- void **clear** ()
- void **resize** (size_t new_size, const T &value={})
- void **reserve** (size_t new_capacity)
- void **erase** (size_t index)
- void **insert** (size_t index, const T &value)
- T & **at** (const size_t index)
- void **shrink_to_fit** ()
- void **swap** (Vector< T > &other)
- T & **front** () const
- T & **back** () const
- size_t **get_size** () const
- size_t **get_capacity** () const
- T * **begin** () const
- T * **end** () const
- T * **get_data** () const
- bool **empty** () const
- size_t **max_size** () const

The documentation for this class was generated from the following file:

- vector.hpp

Chapter 6

File Documentation

6.1 vector.hpp

```
00001 #ifndef VECTOR_HPP
00002 #define VECTOR_HPP
00003 #include <memory>
00004 #include <limits>
00005
00006 template <typename T>
00007 class Vector
00008 {
00009     size_t size;
00010     size_t capacity;
00011     std::unique_ptr<T[]> data;
00012
00013 public:
00014     Vector() : size(0), capacity(1), data(new T[1]) {}
00015     ~Vector()
00016     {
00017         size = 0;
00018         capacity = 0;
00019         data.release();
00020     }
00021
00022     Vector(const Vector<T> &other)
00023     {
00024         size = other.size;
00025         capacity = other.capacity;
00026         data.reset(new T[capacity]);
00027         for (size_t i = 0; i < size; i++)
00028         {
00029             data[i] = other.data[i];
00030         }
00031     }
00032
00033     Vector<T> &operator=(const Vector<T> &other)
00034     {
00035         if (this != &other)
00036         {
00037             size = other.size;
00038             capacity = other.capacity;
00039             data.reset(new T[capacity]);
00040             for (size_t i = 0; i < size; i++)
00041             {
00042                 data[i] = other.data[i];
00043             }
00044         }
00045         return *this;
00046     }
00047
00048     Vector(Vector<T> &&other) noexcept
00049     {
00050         size = std::move(other.size);
00051         capacity = std::move(other.capacity);
00052         data = std::move(other.data);
00053     }
00054
00055     Vector<T> &operator=(Vector<T> &&other) noexcept
00056     {
00057         if (this != &other)
```

```
00058         {
00059             size = std::move(other.size);
00060             capacity = std::move(other.capacity);
00061             data = std::move(other.data);
00062         }
00063         return *this;
00064     }
00065
00066     T &operator[](const size_t index)
00067     {
00068         if (index < size)
00069         {
00070             return data[index];
00071         }
00072
00073         throw std::out_of_range("operator[] index out of range");
00074     }
00075
00076     bool operator==(const Vector<T> &other)
00077     {
00078         if (size != other.size)
00079         {
00080             return false;
00081         }
00082
00083         for (size_t i = 0; i < size; i++)
00084         {
00085             if (data[i] != other.data[i])
00086             {
00087                 return false;
00088             }
00089         }
00090         return true;
00091     }
00092
00093     bool operator!=(const Vector<T> &other)
00094     {
00095         if (size != other.size)
00096         {
00097             return true;
00098         }
00099
00100         for (size_t i = 0; i < size; i++)
00101         {
00102             if (data[i] != other.data[i])
00103             {
00104                 return true;
00105             }
00106         }
00107         return false;
00108     }
00109
00110     void assign(const size_t count, const T &value)
00111     {
00112         this->resize(count);
00113         for (size_t i = 0; i < count; i++)
00114         {
00115             data[i] = value;
00116         }
00117     }
00118
00119     void push_back(const T &value)
00120     {
00121         if (size >= capacity)
00122         {
00123             capacity = (capacity == 0) ? 1 : capacity * 2;
00124             std::unique_ptr<T[]> new_data(new T[capacity]);
00125             for (size_t i = 0; i < size; i++)
00126             {
00127                 new_data[i] = data[i];
00128             }
00129             data.swap(new_data);
00130         }
00131         data[size++] = value;
00132     }
00133
00134     T pop_back()
00135     {
00136         if (size > 0)
00137         {
00138             return data[--size];
00139         }
00140         throw std::out_of_range("pop_back() called on empty vector");
00141     }
00142
00143     void clear()
00144     {
```

```
00145         size = 0;
00146         capacity = 1;
00147         data.reset(new T[capacity]);
00148     }
00149
00150     void resize(size_t new_size, const T &value = {})
00151     {
00152         if (new_size > capacity)
00153         {
00154             capacity = new_size;
00155         }
00156
00157         std::unique_ptr<T[]> new_data(new T[capacity]);
00158         for (size_t i = 0; i < size; i++)
00159         {
00160             new_data[i] = data[i];
00161         }
00162         for (size_t i = size; i < new_size; i++)
00163         {
00164             new_data[i] = value;
00165         }
00166
00167         size = new_size;
00168         data.swap(new_data);
00169     }
00170
00171     void reserve(size_t new_capacity)
00172     {
00173         if (new_capacity < size)
00174         {
00175             throw std::out_of_range("reserve() new capacity is less than current size");
00176         }
00177
00178         std::unique_ptr<T[]> new_data(new T[new_capacity]);
00179         for (size_t i = 0; i < size; i++)
00180         {
00181             new_data[i] = data[i];
00182         }
00183         data.swap(new_data);
00184         capacity = new_capacity;
00185     }
00186
00187     void erase(size_t index)
00188     {
00189         if (index < size)
00190         {
00191             for (size_t i = index; i < size - 1; i++)
00192             {
00193                 data[i] = data[i + 1];
00194             }
00195             size--;
00196         }
00197         else
00198         {
00199             throw std::out_of_range("erase() index out of range");
00200         }
00201     }
00202
00203     void insert(size_t index, const T &value)
00204     {
00205         if (index < size)
00206         {
00207             if (size >= capacity)
00208             {
00209                 capacity = (capacity == 0) ? 1 : capacity * 2;
00210                 std::unique_ptr<T[]> new_data(new T[capacity]);
00211                 for (size_t i = 0; i < size; i++)
00212                 {
00213                     new_data[i] = data[i];
00214                 }
00215                 data.swap(new_data);
00216             }
00217
00218             for (size_t i = size - 1; i >= index; i--)
00219             {
00220                 data[i + 1] = data[i];
00221             }
00222             size++;
00223             data[index] = value;
00224         }
00225         else
00226         {
00227             throw std::out_of_range("insert() index out of range");
00228         }
00229     }
00230
00231     T &at(const size_t index)
```

```
00232     {
00233         if (index < size)
00234         {
00235             return data[index];
00236         }
00237 
00238         throw std::out_of_range("at() index out of range");
00239     }
00240 
00241     void shrink_to_fit()
00242     {
00243         std::unique_ptr<T[]> new_data(new T[size]);
00244         for (size_t i = 0; i < size; i++)
00245         {
00246             new_data[i] = data[i];
00247         }
00248         data.swap(new_data);
00249         capacity = size;
00250     }
00251 
00252     void swap(Vector<T> &other)
00253     {
00254         std::swap(size, other.size);
00255         std::swap(capacity, other.capacity);
00256         data.swap(other.data);
00257     }
00258 
00259     T &front() const
00260     {
00261         if (size > 0)
00262         {
00263             return data[0];
00264         }
00265         throw std::out_of_range("front() called on empty vector");
00266     }
00267 
00268     T &back() const
00269     {
00270         if (size > 0)
00271         {
00272             return data[size - 1];
00273         }
00274         throw std::out_of_range("back() called on empty vector");
00275     }
00276 
00277     size_t get_size() const
00278     {
00279         return size;
00280     }
00281 
00282     size_t get_capacity() const
00283     {
00284         return capacity;
00285     }
00286 
00287     T *begin() const
00288     {
00289         return data.get();
00290     }
00291 
00292     T *end() const
00293     {
00294         return data.get() + size;
00295     }
00296 
00297     T *get_data() const
00298     {
00299         return data.get();
00300     }
00301 
00302     bool empty() const
00303     {
00304         return size == 0;
00305     }
00306 
00307     size_t max_size() const
00308     {
00309         return std::numeric_limits<size_t>::max();
00310     }
00311 };
00312 #endif // VECTOR_HPP
```

Index