

**KAUNO TECHNOLOGIJOS UNIVERSITETAS**

**INFORMATIKOS FAKULTETAS**

**Mangirdas Kazlauskas**

**„Iš dainos žodžių neišmesi“ – lygiagreti dainos žodžių paieška**

Individualus darbas

**Darbą priėmė:**

doc. dr. Romas Marcinkevičius

**KAUNAS, 2016**

Turinio lentelė

[1 Užduotis 3](#_Toc469687035)

[2 Užduoties analizė ir sprendimo metodas 3](#_Toc469687036)

[3 Programos aprašymas 5](#_Toc469687037)

[3.1 Trumpi klasių, duomenų struktūrų ir metodų aprašai 5](#_Toc469687038)

[3.2 Proceso realizavimo aprašas. 7](#_Toc469687039)

[3.3 Panaudotų LP priemonių aprašas 12](#_Toc469687040)

[4 Programos pagrindinių dalių tekstai su komentarais 13](#_Toc469687041)

[5 Testavimas ir programos instaliavimo bei vykdymo instrukcija 21](#_Toc469687042)

[5.1 Testavimas 21](#_Toc469687043)

[5.2 Programos instaliavimo instrukcija 22](#_Toc469687044)

[5.3 Vykdymo instrukcija 22](#_Toc469687045)

[6 Vykdymo laiko kitimo tyrimas 23](#_Toc469687046)

[7 Išvados 31](#_Toc469687047)

[8 Literatūra 31](#_Toc469687048)

# Užduotis

Dainos kūrime dalyvauja dvi pusės: prodiuseriai ir vadybininkas. Prodiuseriai kuria dainai žodžius, o vadybininkas reguliuoja prodiuserius, tikrina jų sukurtus žodžius. Šio darbo bei užduoties tikslas – įsitikinti, ar iš prodiuserių kuriamų žodžių įmanoma sukurti norimos dainos tekstą.

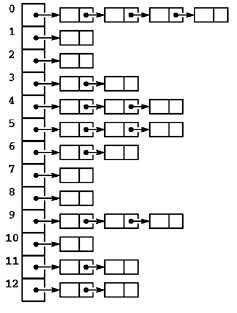
# Užduoties analizė ir sprendimo metodas

Tokio tipo užduotis, kuri aprašyta pirmame skyriuje, sprendžiama remiantis žodžių paieška tekste. Visas užduoties sprendimas susideda iš 4 etapų:

1. **Dainos teksto nuskaitymas.** Prieš vykdant pagrindinius programos veiksmus, nuskaitomas dainos tekstas, kuriame bus ieškoma prodiuserių kuriamų žodžių. Keblumų gali kilti dėl to, kad dainos tekstą sudaro ne tik pavieniai žodžiai, bet ir įvairūs skyrybos ženklai, kurių, žinoma, prodiuseriai nekuria (jie rūpinasi tik žodžių kūrimų), todėl programa turi pasirūpinti, kad skaitomas dainos tekstas būtų išskaidytas į pavienius žodžius. Šiam tikslui naudojamos reguliariosios išraiškos – taisyklių rinkiniai, aprašantys tekstinį šabloną, pagal kurį randamas tam tikras simbolis ar simbolių seka. Dainos žodžių paieškai naudojamas toks šablonas: **[^\\p{Alnum}‘-]+**. Reguliariąją išraišką sudarančių simbolių prasmė:
2. [] – šablone naudojamas vienas iš simbolių, aprašytų tarp šitų skliaustų;
3. **^ -** eilutės pradžios simbolis;
4. \\p{Alnum} – alfabeto simbolis (didžioji arba mažoji raidė) arba skaičius (0-9);
5. ‘ – apostrofo simbolis;
6. **- -** brūkšnio simbolis;
7. + - simbolių seka, esanti tarp [] gali pasikartoti 1 ir daugiau kartų;

Kaip matome, dainos tekste bus ieškoma simbolių seka, susidedanti iš raidžių arba skaičių, apostrofo bei brūkšnio – simbolių seka, atitinkanti bet kurį dainos žodį. Naudojant šią reguliariąją išraišką, dainos tekstas nuskaitomas, o pavieniai žodžiai išsaugojami sąrašo duomenų struktūroje, prieš tai visas žodžio raides paverčiant į mažąsias, nes tam, kad žodžiai būtų lygūs, nereikia, jog kartu sutaptų ir raidžių dydžiai, pvz., jei žodis labas bus sakinio pradžioje, tai jis bus rašomas „Labas“, tačiau vis tiek žodžio reikšmė išlieka ta pati, nepriklausomai nuo to, ar jo pirmoji raidė parašyta iš mažosios ar didžiosios raidės.

1. **Prodiuserių kuriamų žodžių nuskaitymas.** Prodiuserių kuriamų žodžių nuskaitymas yra identiškas dainos teksto nuskaitymui, t.y., prodiuseris taip gali kurti ne pavienius žodžius, bet, pvz., dainos posmą, kuriame gali būti skyrybos ženklų, todėl pavienių žodžių išskyrimui naudojama ta pati reguliarioji išraiška, kuri naudojama ir dainos teksto nuskaitymui. Prodiuserių žodžių nuskaitymas nuo dainos teksto nuskaitymo skiriasi tik tuo, kad prodiuserių gali būti daugiau negu vienas, jie visi žodžius kuria atskiruose tekstiniuose failuose, todėl atskirai kiekvieno prodiuserio sukurti jau išskirti pavieniai žodžiai sudedami į sąrašus (prieš tai raides taip pat paverčiant į mažąsias), o po to visi sąrašai sudedami į vieną bendrą prodiuserių gaminamų žodžių sąrašą, t.y, galutinė prodiuserių sukurtų žodžių struktūra – sąrašų sąrašas (1 pav.).

****

pav. Sąrašų sąrašo struktūra

1. **Prodiuserių sukurtų žodžių paieška dainos tekste.** Kai duomenys nuskaityti ir patalpinti duomenų struktūrose, vykdoma prodiuserių sukurtų žodžių paieška dainos tekste. Kadangi tiek dainos tekstas, tiek prodiuserių sukurti tekstai jau yra suskaidyti į pavienius atskirus žodžius, o jų didžiosios raidės paverstos mažosiomis, lyginant žodžius nereikia atlikti jokių papildomų veiksmų, užtenka tik palyginti, ar prodiuserio žodis yra lygus dainos žodžiui. Norint įsitikinti, ar visi norimos sukurti dainos žodžiai yra surasti, paiešką atitinka tokia veiksmų seka:
2. Iš atsitiktinio prodiuserio sukurtų žodžių struktūros paimamas pirmas jo dar dainos tekste neieškotas žodis;
3. Paimtas žodis ieškomas dainos tekste, paiešką pradedant nuo dainos teksto pradžios (duomenų struktūros, saugančios dainos tekstą, pirmojo elemento);
4. Tikrinama, ar žodis lygus atitinkamam dainos teksto žodžiui;
5. Jei žodžiai sutampa, dainos teksto žodžių duomenų struktūroje pažymima, kad jau atitinkamas dainos žodis yra sukurtas, šio prodiuserio sukurto žodžio paieška dainos tekste nutraukiama. Kitu atveju – imamas sekantis dainos teksto žodis, iš naujo vykdomi 3-4 paieškos veiksmai;
6. **Paieškos nutraukimas bei rezultatų išvedimas.** Prodiuserių sukurtų žodžių paieška dainos tekste yra nutraukiama po to, kai yra sukurtas visas dainos tekstas (rasti visi dainos teksto žodžių atitikmenys tarp prodiuserių sukurtų žodžių) arba tada, kai kuriamos dainos žodžių paieška atlikta su visais prodiuserių sukurtais žodžiais. Po paieškos į ekraną išvedami rezultatai: **„DAINA SUKURTA“** – jei dainą pavyko sukurti (visi dainos žodžiai buvo rasti), priešingu atveju į ekraną išvedama žinutė **„DAINOS SUKURTI NEPAVYKO“**. Taip pat į ekraną išvedama ir paieškos vykdymo trukmė milisekundėmis.

Šiame individualiame darbe bus stengiamasi šią paiešką optimizuoti naudojant lygiagretaus programavimo priemones. Paieška bus optimizuota tokiu būdu: kadangi dainos teksto žodžius kuria ne vienas, o daug prodiuserių, tačiau vadybininkas, kuris rūpinasi teksto surinkimu yra tik vienas, tai prodiuserių kuriami žodžiai bus siunčiami vadybininkui vienu metu iš kelių skirtingų prodiuserių: taip betarpiškai siunčiant žodžius tikimasi, kad daina bus sukurta greičiau nei žodžius nagrinėjant nuosekliai paiešką vykdant nuo vieno prodiuserio žodžių pereinant prie kitų, pvz., galbūt dainai reikalingus teksto žodžius kuria tik vienas prodiuseris, kuris, ne lygiagrečios paieškos atveju, būtų nagrinėjamas paskutinis, todėl lygiagreti paieška šio prodiuserio žodžius surastų ir nagrinėtų greičiau.

# Programos aprašymas

Šiame skyriuje pateikiami trumpi klasių, funkcijų, duomenų struktūrų aprašai. Taip pat aprašomas lygiagrečių procesų veikimas, apibrėžiamas lygiagretaus programavimo priemonės, panaudotos programos kūrimui.

## Trumpi klasių, duomenų struktūrų ir metodų aprašai

1 lentelėje pateikiamos programoje naudojamos klasės, funkcijų, duomenų struktūrų antraštės bei jų aprašymai.

lentelė. Klasės DainosZodis struktūros aprašas

|  |  |
| --- | --- |
| **Klasės pavadinimas:** DainosZodis | |
| **Klasės paskirtis:** Klasė, kurioje saugoma vieną dainos žodį aprašanti struktūra | |
| **Klasės fragmentas** | **Klasės fragmento aprašymas** |
| private final String zodis;  private boolean sukurtas; | Dainos žodis  Loginis kintamasis, nurodantis, ar atitinkamas dainos žodis jau yra sukurtas (rastas tarp prodiuserių kuriamų žodžių) |
| public void keistiSukurtas(); | Keičia dainos žodžio būseną į sukurtą (sukurtas = true); |

lentelė. Klasės Prodiuseris struktūros aprašas

|  |  |
| --- | --- |
| **Klasės pavadinimas:** Prodiuseris | |
| **Klasės paskirtis:** Klasė, aprašanti prodiuserį bei jo veiksmus | |
| **Klasės fragmentas** | **Klasės fragmento aprašymas** |
| private final int id;  private final List<String> sukurtiZodziai; | Prodiuserio proceso identifikacinis numeris  Prodiuserio sukurtų žodžių sąrašas |
| private void siusti(); | Metodas, kuris perduoda prodiuserio sukurtus žodžius vadybininkui |

lentelė. Klasės Vadybininkas struktūros aprašas

|  |  |
| --- | --- |
| **Klasės pavadinimas:** Vadybininkas | |
| **Klasės paskirtis:** Klasė, aprašanti vadybininką bei jo veiksmus | |
| **Klasės fragmentas** | **Klasės fragmento aprašymas** |
| public List<DainosZodis> dainosZodziai = new ArrayList<>();  private final boolean kuria[]; | Dainos žodžių struktūrų sąrašas  Loginių kintamųjų masyvas, i-joje vietoje saugantis logines reikšmes, ar i-tasis prodiuseris vis dar siunčia savo sukurtus žodžius vadybininkui |
| private boolean dainaSukurta(); | Metodas, grąžinantis loginę reikšmę true, jei daina jau yra sukurta |
| private boolean yraProdiuseris(); | Metodas, grąžinantis loginę reikšmę true, jei dar yra vadybininkui žodžius siunčiančių (kuriančių) prodiuserių |
| private void nustatytiKadProdiuserisNebekuria(int id) | Metodas, nustatantis, kad parametru nurodyto id prodiuseris baigė vadybininkui siųsti savo sukurtus žodžius |
| private void ieskotiDainoje(String zodis); | Metodas, tarp dainos žodžių ieškantis parametru zodis nurodyto žodžio |
| private void kurti(); | Metodas, kuriame vadybininkas atlieka paieškos bei paieškos nutraukimo veiksmus |

lentelė. Klasės KazlauskasM\_ID struktūros aprašas

|  |  |
| --- | --- |
| **Klasės pavadinimas:** KazlauskasM\_ID | |
| **Klasės paskirtis:** Pagrindinė individualaus darbo programos klasė | |
| **Klasės fragmentas** | **Klasės fragmento aprašymas** |
| private static String dainosPavadinimas = Konstantos.DAINA\_BOHEMIAN\_RHAPSODY;  private static int giju\_sk = Konstantos.PRODIUSERIU\_SK;  private static boolean generuoti = true;  private static int generavimo\_budas = 2; | Dainos pavadinimas  Gijų (prodiuserių) skaičius  Loginė reikšmė, nurodanti, ar reikia generuoti prodiuserių kuriamus žodžius  Kintamasis, nurodantis prodiuserių žodžių generavimo būdą, kur 1 - žodžiai suskaidomi blokais, pradedant pirmuoju po lygiai kiekvienam prodiuseriui (jei lygiai neišeina – n-1 prodiuseriams po lygiai, o n-tajam prodiuseriui – visi likę žodžiai), o 2 - Žodžiai prodiuseriams skirstomi po vieną, pradedant pirmuoju, skirstoma, kol dar yra žodžių. |
| private static void duomenuGeneravimas(String daina, int n, int generavimo\_budas); | Funkcija, generuojanti prodiuserių žodžius (parametrai tie patys, kurie aprašyti aukščiau) |
| private static void skaitytiDainosTeksta(String dainosPavadinimas, List<DainosZodis> dainosZodziai) | Dainos teksto skaitymo funkcija, kur parametras dainosPavadinimas nurodo dainos pavadinimą, o žodžiai išsaugojami sąraše dainosZodziai |
| private static void skaitytiProdiuseriuZodzius(List<List<String>> visuProdiuseriuZodziai, String dainosPavadinimas, int n) | Prodiuserių kuriamų žodžių skaitymo funkcija, kur prodiuserių žodžiai saugojami sąraše visuProdiuseriuZodziai, dainosPavadinimas nurodo dainos pavadinimą, o parametras n nurodo prodiuserių skaičių |
| private static void skaitytiParametrus(String args[]) | Metodas, skaitantis bei validuojantis parametrus, kai programa vykdoma komandinėje eilutėje |

## Proceso realizavimo aprašas.

Dainos kūrimo programoje naudojami dviejų tipų procesai: prodiuserio ir valdytojo. Jų atliekamos funkcijos skiriasi, tačiau bendra realizavimo struktūra yra panaši. Procesai realizuojami naudojant JavaCSP paketą, skirtą JAVA nuoseklių procesų komunikavimui. Tam, kad procesai naudotų šį paketą, realizuojamos klasės turi naudoti CSProcess sąsają. Taip pat bendras šios sąsajos naudojimo panašumas – klasės, naudojančios CSProcess sąsają, privalo užkloti metodą run(), kuris aprašo realizuotame procese vykdomus veiksmus. Šią sąsają naudojantys procesai gali sąveikauti tarpusavyje, naudodamiesi kanalais.

lentelė. Prodiuserio proceso realizacija

|  |
| --- |
| class Prodiuseris implements CSProcess |
| class Prodiuseris implements CSProcess{  // Prodiuserio identifikacinis numeris  private final int id;  // Prodiuserio sukurtų žodžių sąrašas  private final List<String> sukurtiZodziai;  // Kanalas, skirtas perduoti žodžius vadybininko procesui  private final ChannelOutput kanalas\_zodziai\_out;  // Kanalas, skirtas perduoti prodiuserio būseną vadybininko procesui  private final ChannelOutputInt kanalas\_kuria\_out;  // Kanalas, skirtas gauti žinutei iš vadybininko proceso, ar reikia  // baigti darbą  private final AltingChannelInputInt kanalas\_stabdyti\_in;    // Konstruktorius su parametrais  // @param id - prodiuserio identifikacinis numeris  // @param sukurtiZodziai - prodiuserio sukurtų žodžių sąrašas  // @param kanalas\_zodziai\_out - kanalas, skirtas perduoti žodžius vadybininko procesui  // @param kanalas\_kuria\_out - kanalas, skirtas perduoti prodiuserio būseną vadybininko procesui  // @param kanalas\_stabdyti\_in - kanalas, skirtas gauti žinutei iš vadybininko proceso, ar reikia  // baigti darbą  public Prodiuseris(int id,  List<String> sukurtiZodziai,  ChannelOutput kanalas\_zodziai\_out,  ChannelOutputInt kanalas\_kuria\_out,  AltingChannelInputInt kanalas\_stabdyti\_in){  this.id = id;  this.sukurtiZodziai = sukurtiZodziai;  this.kanalas\_zodziai\_out = kanalas\_zodziai\_out;  this.kanalas\_kuria\_out = kanalas\_kuria\_out;  this.kanalas\_stabdyti\_in = kanalas\_stabdyti\_in;  }    // Metodas, skirtas siųsti prodiuserio proceso pagamintus žodžius vadybininkui  private void siusti(){  // Einama per visus prodiuserio sukurtus žodžius  for (String zodis : sukurtiZodziai){  // Tikrinama, ar negauta žinutė iš vadybininko, kad reikia nutraukti  // gamybos procesą  if (kanalas\_stabdyti\_in.pending()){  // Jei žinutė gauta - darbas nutraukiamas  if (kanalas\_stabdyti\_in.read() == -1){  break;  }  }  // Kanalu išsiunčiamas prodiuserio sukurtas žodis vadybininkui  kanalas\_zodziai\_out.write(zodis);  }  // Pabaigus siųsti visus prodiuserio sukurtus žodžius, vadybininkui  // išsiunčiama žinutė, kad prodiuserio procesas baigė darbą  kanalas\_kuria\_out.write(id);  }    // Užklotas CSProcess sąsajos metodas run()  @Override  public void run(){  siusti();  }  } |

Prodiuserio proceso realizavimo klasė saugo 5 kintamuosius:

* Id – prodiuserio proceso identifikacinis numeris;
* sukurtiZodziai – sąrašo duomenų struktūra, sauganti prodiuserio sukurtus žodžius;
* kanalas\_zodziai\_out – kanalas, kuriuo žodžiai bus perduodami vadybininko procesui;
* kanalas\_kuria\_out – kanalas, kuriuo prodiuseris vadybininko procesui perduos žinutę, jog baigė darbą;
* kanalas\_stabdyti\_in – kanalas, kuriuo prodiuseris gauna žinutę iš vadybininko proceso, jei reikia stabdyti kūrimo (žodžių siuntimo vadybininkui) procesą

Procese naudojamas metodas siusti(), kuriame pirmiausia patikrinama, ar vadybininko procesas neatsiuntė žinutės, jog reikia nutraukti žodžių kūrimo darbą. Jei žinutė neatsiųsta, tada imamas paskutinis vadybininkui neišsiųstas žodis iš sąrašo sukurtiZodziai ir kanalu kanalas\_zodziai\_out siunčiamas vadybininko procesui. Kai visi prodiuserio žodžiai išsiųsti, vadybininko procesui siunčiamas signalas kanalu kanalas\_kuria\_out, kuris nurodo, kad prodiuseris žodžių kūrimo darbą jau baigė.

Metodas run() paleidžia metode apibrėžtus procesus veiksmus lygiagrečiai su kitais metodais, užregistruotais tame pačiame Parallel objekte.

lentelė. Vadybininko proceso realizacija

|  |
| --- |
| class Vadybininkas implements CSProcess |
| class Vadybininkas implements CSProcess{  // Sąrašas, saugantis visų dainos žodžių struktūras  public List<DainosZodis> dainosZodziai = new ArrayList<>();  // Masyvas, saugantis pagal indeksą atitinkamų prodiuserių būsenas  // jei kuria[id] = true - indekso id prodiuseris vis dar kuria žodžius  private final boolean kuria[];  // Kanalas, skirtas gauti žodžius iš prodiuserių procesų  private final AltingChannelInput kanalas\_zodziai\_in;  // Kanalas, skirtas gauti prodiuserių procesų būsenas iš prodiuserių  private final AltingChannelInputInt kanalas\_produseriai\_kuria\_in;  // Kanalų masyvas, skirtas išsiųsti žinutę prodiuseriams, kad jie baigtų darbą  // (nustotų kurti ir siųsti vadybininkui kuriamus žodžius)  private final One2OneChannelInt[] kanalas\_stabdyti\_out;    // Konstruktorius su parametrais  // @param dainosZodziai - dainos žodžių struktūrų sąrašas  // @param kanalas\_zodziai\_in - kanalas, skirtas gauti žodžius iš prodiuserių procesų  // @param kanalas\_prodiuseriai\_kuria\_in - kanalas, skirtas gauti prodiuserių procesų būsenas iš prodiuserių  // @param kanalas\_stabdyti\_out - kanalų masyvas, skirta stabdyti prodiuserių procesus  // @param giju\_sk - prodiuserių procesų (gijų) skaičius  public Vadybininkas(List<DainosZodis> dainosZodziai,  AltingChannelInput kanalas\_zodziai\_in,  AltingChannelInputInt kanalas\_prodiuseriai\_kuria\_in,  One2OneChannelInt[] kanalas\_stabdyti\_out,  int giju\_sk){  this.dainosZodziai = dainosZodziai;  kuria = new boolean[giju\_sk];  // Visos prodiuserių būsenos nustatomos į kuriančius (kuria[id] = true)  for (int i = 0; i < giju\_sk; i++){  kuria[i] = true;  }  this.kanalas\_zodziai\_in = kanalas\_zodziai\_in;  this.kanalas\_produseriai\_kuria\_in = kanalas\_prodiuseriai\_kuria\_in;  this.kanalas\_stabdyti\_out = kanalas\_stabdyti\_out;  }    // Funkcija, grąžinanti true, jei daina jau yra sukurta  // return true - jei sukurta, return false - jei daina vis dar turi nesukurtų žodžių  private boolean dainaSukurta(){  // Ciklas eina per visas sąrašo dainosZodziai žodžių struktūras  for (DainosZodis dz : dainosZodziai){  if (!dz.sukurtas()){  // Jei bent vienas dainos žodis nesukurtas, grąžinama false  return false;  }  }  // Grąžinama true, jei nerastas nė vienas nesukurtas dainos žodis  return true;  }    // Funkcija, grąžinanti true, jei vis dar yra dainos žodžius kuriančių  // (dainos žodžius siunčiančių) prodiuserio procesų  // return true - jei randamas bent vienas žodžius kuriantis prodiuseris  // return false - jei nėra nė vieno kuriančio prodiuserio  private boolean yraProdiuseris(){  // Ciklas eina per prodiuserių būsenų masyvą  for (boolean k : kuria){  // Jei randama bent viena būsena - true, reiškia, kad yra bent  // vienas dainos žodžius siunčiantis prodiuserio procesas  if(k){  return true;  }  }  // Grąžinama false, jei visos prodiuserių procesų būsenos - false  return false;  }    // Funkcija, nustatanti parametru nurodyto id prodiuserio proceso būseną  // @param id - prodiuserio proceso identifikacinis numeris  private void nustatytiKadProdiuserisNebekuria(int id){  kuria[id] = false;  }    // Funkcija, sąraše dainosZodziai ieškanti parametru nurodyto dainos žodžio  // @param zodis - dainoje ieškomas prodiuserio proceso atsiųstas dainos žodis  private void ieskotiDainoje(String zodis){  // Ciklas eina per sąrašą dainosZodziai  for (DainosZodis dz : dainosZodziai){  // Tikrinama, ar žodžiai sutampa ir ar atitinkamas dainos teksto  // žodis dar nėra sukurtas  if (Objects.equals(dz.imtiZodi(), zodis) && !dz.sukurtas()){  // Jei tinkamas žodis randamas dainos tekste, jo būsena pakeičiama  // į sukurtą  dz.keistiSukurtas();  return;  }  }  }    // Funkcija, vykdanti dainos kūrimo veiksmus, t.y., tikrina, ar prodiuserių  // procesai siunčia žodžius, ar prodiuserių procesai kanalu siunčia žinutę,  // jog jau baigė darbą bei kanalu siunčia žinutes prodiuserių procesams, jog  // daina jau sukurta ir reikia nutraukti žodžių kūrimo darbą  private void kurti(){  while(true){  // Tikrinama, ar prodiuserių žodžių siuntimo kanale yra nenuskaitytų  // žodžių  if(kanalas\_zodziai\_in.pending()){  // Žodis skaitomas iš kanalo  Object duomenys = kanalas\_zodziai\_in.read();  String zodis = (String) duomenys;  // Žodis ieškomas kuriamos dainos tekste  ieskotiDainoje(zodis);  }  // Tikrinama, ar kažkuris prodiuseris kanalu siunčia būseną, jog  // baigė dainos žodžių kūrimą  if(kanalas\_produseriai\_kuria\_in.pending()){  int id = kanalas\_produseriai\_kuria\_in.read();  nustatytiKadProdiuserisNebekuria(id);  }  // Tikrinama, ar daina jau sukurta  if(dainaSukurta()){  // Jei daina sukurta, prodiuserių procesams siunčama žinutė,  // kad šie nustotų kanalu siųsti žodžius  for (int i = 0; i < kuria.length; i++){  kanalas\_stabdyti\_out[i].out().write(-1);  }  break;  }  // Tikrinama sąlyga, ar vis dar yra žodžius kuriančių prodiuserių  // ir ar žodžių siuntimo kanale yra dar nenuskaitytų žodžių  else if(!yraProdiuseris() && !kanalas\_zodziai\_in.pending()){  break;  }  }  }    // Užklotas CSProcess sąsajos metodas run()  @Override  public void run(){  kurti();  // Spausdinami rezultatai pagal tai, ar dainą pavyko sukurti, ar ne  if (dainaSukurta()){  System.out.println("Dainos kūrimo pabaiga.");  System.out.println("\nRezultatas: DAINA SUKURTA");  }  else{  System.out.println("Dainos kūrimo pabaiga.");  System.out.println("\nRezultatas: DAINOS SUKURTI NEPAVYKO");    }  }  } |

Vadybininko proceso realizavimo klasė saugo 5 kintamuosius:

* dainosZodziai – sąrašas, saugantis visų dainos žodžių struktūras;
* kuria[] – masyvas, kuriame saugomos prodiuserių būsenos, nurodančios, ar atitinkamas id vis dar kuria žodžius dainai;
* kanalas\_zodziai\_in – kanalas, kuriuo vadybininkas gauna žodžius ir prodiuserių procesų;
* kanalas\_prodiuseriai\_kuria\_in – kanalas, kuriuo vadybininkas gauna žinutes ir prodiuserių procesų, kai šie išsiunčia visus savo sukurtus žodžius;
* kanalas\_stabdyti\_out – kanalas, kuriuo vadybininkas siunčia žinutę visiems prodiuserių procesams, jog šie nutrauktų darbą, nes daina jau sukurta.

Tam kad vadybininko procesas įsitikintų ar daina yra sukurta, naudojamas metodas dainaSukurta(), grąžinantis true, jei daina jau yra sukurta (visi dainos žodžiai jau yra sukurti prodiuserių).

Taip pat naudojamas metodas yraProdiuseris(), kuris grąžina true, jei dar yra likusių dainos žodžius kuriančių prodiuserių (prodiuserių procesų, kurie vadybininkui vis dar siunčia žodžius, naudojamus paieškai dainoje).

Metodas nustatytiKadProdiuserisNebekuria(int id) nustato parametru nurodyto id prodiuserio proceso būseną į false (numerio id prodiuserio procesas dainos žodžių nebekuria).

Vadybininko procesas naudoja metodą ieskotiDainoje(String zodis), kuris tarp visų sąrašo dainosZodziai ieško parametru zodis nurodyto žodžio, o jei jį randa – pakeičia dainos žodžio statusą į sukurtą.

Metodas kurti() vykdo šiuos veiksmus: tikrina, ar prodiuserių procesai vis dar siunčia žodžius, kuriuos reikia ieškoti dainoje (jei taip, pasiima atsiųstą prodiuserio žodį ir jo ieško dainoje). Vėliau metodas tikrina, ar kažkuris prodiuseris kanalu siunčia žinutę, jog baigė kurti žodžius (jei taip, žinute gauto id prodiuserio proceso būsena nustatoma į nekuriančią, t.y. kuria[id] = false). Taip pat šis metodas rūpinasi ir valdytojo proceso veikimo pabaigas sąlyga: jei daina sukurta, visiems prodiuseriams išsiunčiama žinutė, kad daugiau dainos žodžių nebekurtų, o jei tenkinama tokia sąlyga, kad daugiau nebėra dainos žodžius kuriančių prodiuserių procesų ir žodžių kanale nebeliko jokių nenuskaitytų žodžių paieškai, metodas taip pat baigia savo darbą.

Metodas run() paleidžia metode apibrėžtus procesus veiksmus lygiagrečiai su kitais metodais, užregistruotais tame pačiame Parallel objekte.

## Panaudotų LP priemonių aprašas

Dainos kūrimo programai realizuoti buvo naudotos JavaCSP paketo lygiagretaus programavimo priemonės. metodai:

1. **Parallel** – klasė, skirta lygiagrečiai paleidžiamų CSProcess procesų apjungimui. Programoje iš pradžių yra sukuriamas naujas Parallel klasės objektas, o vėliau prie jo prijungiami CSProcess sąsajos procesai. Parallel klasė, turinti n gijų, pradėdama lygiagretų programos ar jos dalies darbą, paleidžia visus toje klasėje užregistruotus (prie Parallel klasės objekto pridėtus) procesus (n-1 gija) n-tąją giją pasilikdamas gijų kontrolei;
2. **addProcess** – Parallel klasės metodas, kuris prideda CSProcess procesą prie Parallel objekto. Pridėti procesai paleidžiami tada, kai iškviečiamas Parallel klasės metodas run();
3. **run()** – Parallel klasės metodas, kuris paleidžia visus procesus, pridėtus prie Parallel klasės objekto, lygiagrečiai. Metodas baigia darbą tik tada, kai savo darbą baigia visi prie Parallel klasės objekto pridėti procesai;

Duomenų mainai tarp CSProcess procesų vyksta kanalais. Programos realizavimui naudojami trijų tipų kanalai (sąsajos):

1. **Any2OneChannel** – kanalo sąsaja, kuri yra naudojama tada, kai yra daug procesų-siuntėjų (procesų, siunčiančių duomenis) ir vienas procesas gavėjas. Kanalu gali būti siunčiami objektai (struktūros objektas, klasės objektas, String, Integer ir pan.);
2. **Any2OneChannelInt** - kanalo sąsaja, kuri yra naudojama tada, kai yra daug procesų-siuntėjų (procesų, siunčiančių duomenis) ir vienas procesas gavėjas. Kanalu gali būti siunčiami tik sveikojo tipo skaičiai;
3. **One2OneChannelInt -** kanalo sąsaja, kuri neleidžia kanalu naudotis keliems siuntėjams ar gavėjams – kanalu duomenys siunčiami iš vieno siuntėjo vienam gavėjui. Kanalu gali būti siunčiami tik sveikojo tipo skaičiai;
4. **AltingChannelInputInt** – klasė, leidžianti CSProcess procesui iš kanalo pasirinkti sveikuosius skaičius, kuriuos nori nuskaityti. Tiesa, šioje programos realizacijoje alternatyvos naudojamos nebuvo. Klasė paveldi **ChannelInputInt** sąsają. Klasė savyje turi metodą **pending()**, kuris skirtas duomenų būsenai kanale patikrinti, t.y., grąžina reikšmę true, jei kanale yra nenuskaitytų duomenų;
5. **ChannelOutput** – sąsaja, apibrėžianti duomenų objektų siuntimą kanalais. Šios sąsajos kintamieji negali skaityti iš kanalo (bus metama klaida), taip apsisaugant nuo netinkamų duomenų mainų kanale;
6. **ChannelOutputInt** – sąsaja, atitinkanti sąsają **ChannelOutput,** bet ji apibrėžia tik sveikųjų skaičių siuntimą kanalu;

# Programos pagrindinių dalių tekstai su komentarais

lentelė. Programos išeities kodas

|  |
| --- |
| **Programos išeities kodas su komentarais** |
| /\*  Mangirdas Kazlauskas IFF-4/1  Individualus darbas  Programa: "Iš dainos žodžių neišmesi"  \*/  package id;  import java.io.File;  import java.io.IOException;  import java.io.PrintWriter;  import java.util.ArrayList;  import java.util.List;  import java.util.Objects;  import java.util.Scanner;  import java.util.regex.Pattern;  import org.jcsp.lang.\*;  import org.jcsp.util.InfiniteBuffer;  import org.jcsp.util.ints.InfiniteBufferInt;  //Konstantų klasė, naudojama pradinėms duomenų reikšmės (demonstracijai)  class Konstantos{  // Dainos Bohemian Rhapsody pavadinimas  public static final String DAINA\_BOHEMIAN\_RHAPSODY = "Bohemian Rhapsody";  // Dainos test pavadinimas  public static final String DAINA\_TEST = "test";  // Gijų (prodiuserių) skaičius  public static final int PRODIUSERIU\_SK = 256;  }  // Klasė, skirta saugoti vienos dainos žodžio struktūrai  class DainosZodis{  { ... }  }  // CSProcess sąsajos proceso klasė, aprašanti prodiuserį  class Prodiuseris implements CSProcess{  { ... }  }  // CSProcess sąsajos proceso klasė, aprašantį vadybininką  class Vadybininkas implements CSProcess{  { ... }  }  // Pagrindinė programos klasė  public class KazlauskasM\_ID {  // Kintamieji, kurie pagal nutylėjima priskiriami iš karto  private static String dainosPavadinimas = Konstantos.DAINA\_BOHEMIAN\_RHAPSODY;  private static int giju\_sk = Konstantos.PRODIUSERIU\_SK;  private static boolean generuoti = true;  private static int generavimo\_budas = 2;  // Programos pagrindinis metodas  // @param args - komandinėje eilutėje įvestų parametrų masyvas  public static void main(String[] args) {  // Skaitomi programos naudotojo įvesti parametrai  //skaitytiParametrus(args);  // Į ekraną išvedama informaciją apie vykdomos programos pagrindinius  // parametrus  System.out.println("Pradedama vykdyti programa su šiais duomenimis:");  System.out.format(" Dainos pavadinimas: %s\n", dainosPavadinimas);  System.out.format(" Gijų skaičius: %d\n", giju\_sk);  System.out.format(" Ar vykdomas prodiuserių kuriamų žodžių generavimas?: %s\n", (generuoti) ? "Taip" : "Ne");  if(generuoti) System.out.format(" Prodiuserių kuriamų žodžių generavimo būdas: %s\n", (generavimo\_budas == 1) ? "Žodžiai suskaidomi blokais" : "Žodžiai prodiuseriams skirstomi po vieną");  // Sukuriamas naujas Parallel objektas, neturintis jokių gijų  Parallel visiProcesai = new Parallel();    // Sukuriamas kanalas, skirtas žodžių siuntimui vadybininko procesui  Any2OneChannel kanalas\_zodziaiVadybininkui = Channel.any2one(new InfiniteBuffer());  // Sukuriamas kanalas, skirtas prodiuserių procesų būsenų siuntimui  // vadybininko procesui  Any2OneChannelInt kanalas\_kuria = Channel.any2oneInt(new InfiniteBufferInt());  // Sukuriamas kanalų masyvas, skirtas vadybininko procesui siųsti žinutes  // prodiuserio procesams, kad šie baigtų darbą  One2OneChannelInt[] kanalas\_stabdyti = Channel.one2oneIntArray(giju\_sk, new InfiniteBufferInt());    // Generuojami duomenys (prodiuserių kuriami žodžiai)  if (generuoti) duomenuGeneravimas(dainosPavadinimas, giju\_sk, generavimo\_budas);    // Sukuriami sąrašai, skirti saugoti dainos bei prodiuserių kuriamiems  // žodžiams  List<DainosZodis> dainosZodziai = new ArrayList<>();  List<List<String>> visuProdiuseriuZodziai = new ArrayList<>();    // Skaitomas dainos tekstas  skaitytiDainosTeksta(dainosPavadinimas, dainosZodziai);  // Skaitomi prodiuserių kuriami žodžiai  skaitytiProdiuseriuZodzius(visuProdiuseriuZodziai, dainosPavadinimas, giju\_sk);    // Prie Parallel objekto pridedamas vadybininko procesas  visiProcesai.addProcess(  new Vadybininkas(  dainosZodziai,  kanalas\_zodziaiVadybininkui.in(),  kanalas\_kuria.in(),  kanalas\_stabdyti,  giju\_sk  )  );  // Prie Parallel objekto pridedami prodiuserių procesai  for (int i = 0; i < giju\_sk; i++){  visiProcesai.addProcess(  new Prodiuseris(  i,  visuProdiuseriuZodziai.get(i),  kanalas\_zodziaiVadybininkui.out(),  kanalas\_kuria.out(),  kanalas\_stabdyti[i].in()  )  );  }  System.out.println("\nVykdomas dainos kūrimas...");  // Pradedamas matuoti dainos kūrimo laikas  long start = System.currentTimeMillis();  visiProcesai.run();  // Baigiamas matuoti dainos kūrimo laikas  long finish = System.currentTimeMillis();  System.out.format("Žodžių paieška dainos kūrimui (dainos kūrimo procesas) užtruko: %d ms\n", (finish-start));  }  // Duomenų generavimo funkcija  // @param daina - dainos pavadinimas  // @param n - prodiuserių (gijų) skaičius  // @param - prodiuserių dainos žodžių generavimo būdas  private static void duomenuGeneravimas(String daina, int n, int generavimo\_budas){  List<DainosZodis> dainosZodziai = new ArrayList<>();  // Apibrėžiamas regex šablonas žodžių išskyrimui iš teksto  Pattern regex = Pattern.compile("[^\\p{Alnum}'-]+");  String zodis;  // Nurodomas katalogas, kuriame bus generuojami prodiuserių kuriami žodžiai  File katalogas = new File("duomenys/" + daina + "/zodziai/");  // Jei toks katalogas jau egzistuoja, ištrinami visi ten esantys failai  if(katalogas.exists()){  for(File file: katalogas.listFiles())  if (!file.isDirectory())  file.delete();  }  // Kitu atveju katalogas sukuriamas naujai  else{  try{  katalogas.mkdir();  }  catch(SecurityException ex){  System.out.println(ex);  System.exit(1);  }  }  // Nurodoma daina, kuria remiantis bus generuojami prodiuserių kuriami  // dainos žodžiai  try (Scanner scanner = new Scanner(new File("duomenys/" + daina + "/daina.txt"))) {  // Ciklas vykdomas, kol nuskaitomi visi dainos žodžiai  while (scanner.hasNext()){  scanner.useDelimiter(regex);  // Paimamas paskutinis nenuskaitytas dainos žodis  zodis = scanner.next();  if (!zodis.equals("")){  // Jei žodis nėra tuščia eilutė, jis pridedamas į dainos žodžių sąrašą  DainosZodis naujas = new DainosZodis(zodis.toLowerCase(), false);  dainosZodziai.add(naujas);  }  }  } catch (Exception ex) {  System.out.println(ex);  System.exit(1);  }  // Apskaičiuojama, kiek žodžių bus skiriama vienam prodiuseriui (kad būtų  // padalinta kiek galima tolygiau)  int zodziu\_sk\_prodiuseriui = dainosZodziai.size()/n;  // Žodžio indeksas, nuo kurio bus pradėta dalinti žodžius prodiuseriams  int dabartinis\_zodis = 0;  // Einama per visus prodiuserius  for (int i = 1; i <= n; i++){  try(PrintWriter writer = new PrintWriter("duomenys/" + daina + "/zodziai/" + i + ".txt")) {  // Jei prodiuserių yra daugiau nei dainos žodžių, tai kiekvienas prodiuseris  // gaus po vieną dainos žodį, tačiau tie patys žodžiai kartosis  if(dainosZodziai.size() < n){  writer.println(dainosZodziai.get(dabartinis\_zodis).imtiZodi());  dabartinis\_zodis = (dabartinis\_zodis + 1) % dainosZodziai.size();  }  // Kitu atveju, žodžiai dalinami prodiuseriams, kad nesikartotų  else{  // Jei pasirinktas pirmasis generavimo būdas (kai žodžiai prodiuseriams  // skiriami blokais nuo pradžios  if(generavimo\_budas == 1){  // Jei žodžiai skiriami ne paskutiniam prodiuseriui, tai  // skiriamas atitinkamas žodžių blokas  if (i != n){  while(dabartinis\_zodis < (zodziu\_sk\_prodiuseriui\*i)){  writer.println(dainosZodziai.get(dabartinis\_zodis).imtiZodi());  dabartinis\_zodis++;  }  }  // Paskutiniam prodiuseriui skiriami visi likę neišdalinti  // žodžiai  else{  while(dabartinis\_zodis < dainosZodziai.size()){  writer.println(dainosZodziai.get(dabartinis\_zodis).imtiZodi());  dabartinis\_zodis++;  }  }  }  // Jei pasirinktas antrasis žodžių generavimo būdas, kai žodžiai  // prodiuseriams skirstomi po vieną, pradedant nuo pradžios  else if (generavimo\_budas == 2){  dabartinis\_zodis = (i-1);  while(dabartinis\_zodis < dainosZodziai.size()){  writer.println(dainosZodziai.get(dabartinis\_zodis).imtiZodi());  dabartinis\_zodis += n;  }  }  }  } catch(IOException ex){  System.out.println(ex);  System.exit(1);  }  }  }  // Funkcija, skaitanti dainos tekstą  // @param dainosPavadinimas - dainos pavadinimas  // @param dainosZodziai - sąrašas, kuriame bus saugomi dainos žodžiai  private static void skaitytiDainosTeksta(String dainosPavadinimas, List<DainosZodis> dainosZodziai){  // Apibrėžiamas regex šablonas žodžių išskyrimui iš teksto  Pattern regex = Pattern.compile("[^\\p{Alnum}'-]+");  String zodis;  // Nurodoma daina, kurios žodžiai bus skaitomi  try (Scanner scanner = new Scanner(new File("duomenys/" + dainosPavadinimas + "/daina.txt"))) {  while (scanner.hasNext()){  scanner.useDelimiter(regex);  // Išskiriamas paskutinis dar neišskirtas dainos teksto žodis  zodis = scanner.next();  //System.out.println(zodis);  if (!zodis.equals("")){  // Sukuriama nauja dainos žodžio struktūra  DainosZodis naujas = new DainosZodis(zodis.toLowerCase(), false);  // Naujas žodis pridedamas prie sąrašo  dainosZodziai.add(naujas);  }  }  } catch (Exception ex) {  System.out.println("Klaida skaitant dainos tesktą. Jei norite sužinoti, kaip naudotis programa, skaitykite programos instaliavimo bei vykdymo instrukciją.");  System.out.println(ex);  System.exit(1);  }  }  // Funkcija, skirta skaityti prodiuserių kuriamus žodžius  // @param visuProdiuseriuZodziai - sąrašų sąrašas, saugantis visų prodiuserių  // kuriamus žodžius  // @param dainosPavadinimas - dainos pavadinimas  // @param n - prodiuserių skaičius  private static void skaitytiProdiuseriuZodzius(List<List<String>> visuProdiuseriuZodziai, String dainosPavadinimas, int n){  // Apibrėžiamas regex šablonas žodžių išskyrimui iš teksto  Pattern regex = Pattern.compile("[^\\p{Alnum}'-]+");  // Einama per visų prodiuserių kuriamų žodžių failus  for (int i = 1; i <= n; i++){  List<String> prodiuserioZodziai = new ArrayList<>();  String zodis;  // Nurodoma konkretaus prodiuserio kuriamų žodžių failas  try (Scanner scanner = new Scanner(new File("duomenys/" + dainosPavadinimas + "/zodziai/" + i + ".txt"))) {  while (scanner.hasNext()){  scanner.useDelimiter(regex);  // Išskiriamas paskutinis nenuskaitytas žodis  zodis = scanner.next();  if (!zodis.equals("")){  // Prodiuserio sukurtas žodis pridedamas į sarašą  prodiuserioZodziai.add(zodis.toLowerCase());  }  }  // Visų atitinkamo prodiuserio sukurtų žodžių sąrašas pridedamas  // prie bendro sąrašo  visuProdiuseriuZodziai.add(prodiuserioZodziai);  } catch (Exception ex) {  System.out.println("Klaida skaitant prodiuserio žodžius. Jei norite sužinoti, kaip naudotis programa, skaitykite programos instaliavimo bei vykdymo instrukciją.");  System.out.println(ex);  System.exit(1);  }  }  }  // Funkcija, skaitanti komandinėje eilutėje nurodytus parametrus  // @para args[] - komandinėje eilutėje nurodyti parametrai  private static void skaitytiParametrus(String args[]){  // Tikrinama, ar nurodytas bent vienas parametras  if (args.length == 0){  System.out.println("Nepateikėte nė vieno parametro. Jei norite sužinoti, kaip naudotis programa, skaitykite programos instaliavimo bei vykdymo instrukciją.");  System.exit(1);  }  else{  // Tikrinamas pirmasis parametras  switch(args[0]){  // Ar pirmasis parametras yra -d (demonstracija)  case "-d": {  // Tikrinama, ar nenurodyta per mažai parametrų (viso turi būti 2)  if(args.length < 2){  System.out.println("Nurodėte per mažai parametrų. Jei norite sužinoti, kaip naudotis programa, skaitykite programos instaliavimo bei vykdymo instrukciją.");  System.exit(1);  }  // Nurodytas procesų skaičiaus parametras verčiamas į sveikojo  // skaičiaus kintamąjį  int sk = Integer.parseInt(args[1]);  // Tikrinama, ar nurodytas procesų skaičius nėra lygus 0  if (sk == 0){  System.out.println("Gijų skaičius turi būti didesnis už 0.");  System.exit(1);  }  // Parametrams nustatomos reikšmės  dainosPavadinimas = Konstantos.DAINA\_BOHEMIAN\_RHAPSODY;  giju\_sk = sk;  generuoti = true;  break;  }  // Ar pirmasis parametras yra -g (generacija)  case "-g": {  // Tikrinama, ar nenurodyta per mažai parametrų (viso turi būti 4)  if(args.length < 4){  System.out.println("Nurodėte per mažai parametrų. Jei norite sužinoti, kaip naudotis programa, skaitykite programos instaliavimo bei vykdymo instrukciją.");  System.exit(1);  }  // Nurodytas procesų skaičiaus parametras verčiamas į sveikojo  // skaičiaus kintamąjį  int sk = Integer.parseInt(args[2]);  // Tikrinama, ar nurodytas procesų skaičius nėra lygus 0  if (sk == 0){  System.out.println("Gijų skaičius turi būti didesnis už 0.");  System.exit(1);  }  // Nurodytas duomenų generavimo būdas verčiamas į sveikojo  // skaičiaus kintamąjį  int budas = Integer.parseInt(args[3]);  // Tikrinama, ar nurodytas generavimo būdas yra 1 arba 2  if(budas <= 0 && budas > 2){  System.out.println("Įvestas netinkamas generavimo būdas. Jei norite sužinoti, kaip naudotis programa, skaitykite programos instaliavimo bei vykdymo instrukciją.");  System.exit(1);  }  // Parametrams nustatomos reikšmės  dainosPavadinimas = args[1];  giju\_sk = sk;  generavimo\_budas = budas;  generuoti = true;  break;  }  // Ar pirmasis parametras yra -p (pasirinktinis)  case "-p": {  // Tikrinama, ar nenurodyta per mažai parametrų (viso turi būti 3)  if(args.length < 3){  System.out.println("Nurodėte per mažai parametrų. Jei norite sužinoti, kaip naudotis programa, skaitykite programos instaliavimo bei vykdymo instrukciją.");  System.exit(1);  }  // Nurodytas procesų skaičiaus parametras verčiamas į sveikojo  // skaičiaus kintamąjį  int sk = Integer.parseInt(args[2]);  // Tikrinama, ar nurodytas procesų skaičius nėra lygus 0  if (sk == 0){  System.out.println("Gijų skaičius turi būti didesnis už 0.");  System.exit(1);  }  // Parametrams nustatomos reikšmės  dainosPavadinimas = args[1];  giju\_sk = sk;  generuoti = false;  break;  }  }  }  }  } |

# Testavimas ir programos instaliavimo bei vykdymo instrukcija

## Testavimas

Testavimas buvo atliekamas pagal iš anksto paruoštus scenarijus, planuojamus gauti rezultatus, tada programa buvo vykdoma pagal paruoštą scenarijų ir tikrinama, ar gauti reikiami rezultatai:

lentelė. Programos testavimo scenarijai ir gauti testų rezultatai

|  |  |
| --- | --- |
| **Scenarijus** | **Rezultatas** |
| **Dainos pavadinimas**: test  **Gijų skaičius**: 1  **Dainos tekstas**: testas  **Prodiuserio failo 1.txt tekstas**: testas  **Tikėtinas rezultatas**: Daina sukurta | Daina sukurta |
| **Dainos pavadinimas**: test  **Gijų skaičius**: 1  **Dainos tekstas**: testas  **Prodiuserio failo 1.txt tekstas**: test  **Tikėtinas rezultatas**: Dainos sukurti nepavyko | Dainos sukurti nepavyko |
| **Dainos pavadinimas**: test  **Gijų skaičius**: 1  **Dainos tekstas**: (nėra jokio teksto dainoje)  **Prodiuserio failo 1.txt tekstas**: test  **Tikėtinas rezultatas**: išvedamas klaidos pranešimas, kad daina neturi žodžių | Išvedamas klaidos pranešimas, kad dainą neturi nė vieno žodžio |
| **Dainos pavadinimas**: test  **Gijų skaičius**: 1  **Dainos tekstas**: test  **Prodiuserio failo 1.txt tekstas**: TEST  **Tikėtinas rezultatas**: Daina sukurta | Daina sukurta |
| **Dainos pavadinimas**: test  **Gijų skaičius**: 5  **Dainos tekstas**: test  **Prodiuserių visuose failuose tekstas**: testas  **Tikėtinas rezultatas**: Dainos sukurti nepavyko | Dainos sukurti nepavyko |
| **Dainos pavadinimas**: test  **Gijų skaičius**: 5  **Dainos tekstas**: test  Pirmuose 4-iuose prodiuserių failuose tekstas „testas“, 5-tojo prodiuserio failo tekstas: test  **Tikėtinas rezultatas**: Daina sukurta | Daina sukurta |
| **Dainos pavadinimas**: test  **Gijų skaičius**: 5  **Dainos tekstas**: test  Pirmuose 4-iuose prodiuserių failuose tekstas „testas“, 5-tojo prodiuserio pirmose 999999 eilutėse tekstas – „testas“, 1000000 eilutėje - test  **Tikėtinas rezultatas**: Daina sukurta | Daina sukurta |
| **Dainos pavadinimas**: test  **Gijų skaičius**: 5  **Dainos tekstas**: pirmose 999999 eilutėse tekstas – „testas“, 1000000 eilutėje – test  Pirmuose 4-iuose prodiuserių failuose tekstas „testas“, 5-tojo prodiuserio pirmose 999999 eilutėse tekstas – „testas“, 1000000 eilutėje - test  **Tikėtinas rezultatas**: Daina sukurta | Daina sukurta |

Kitais atvejais leidžiant programą, kai generacija yra įjungta, mes esame užtikrinti, kad daina visada bus sukurta, nes prodiuserių kuriamų žodžių failai yra tiesiogiai kuriami iš nuskaitytų dainos žodžių, tiesiog išskaidomi dalimis prodiuseriams. Todėl tada, kai generacija yra įjungta, daina visada bus sukuriama – tai buvo įsitikinta atliekant vykdymo laiko tyrimą.

## Programos instaliavimo instrukcija

1. Iš svetainės <https://java.com/en/download/> parsisiųskite JAVA;
2. Parsiųstus failus įdiekite kompiuteryje (sekti instrukcijas JAVA instaliacijos metu);
3. Jūsų darbalaukyje susikurkite naują katalogą „Daina“, kuriame talpinsime bei kursime visus programos vykdymui reikalingus failus;
4. Į sukurtą katalogą įkelkite programos failus: katalogą „lib“, katalogą „duomenys“, kuriame yra dainos failai demonstraciniam programos vykdymui bei programos vykdymo failą „ID.jar“;
5. Katalogo „duomenys“ viduje sukurkite naują katalogą, kurio pavadinimas atitiktų Jūsų kuriamos dainos pavadinimą;
6. Jūsų dainos pavadinimo kataloge sukurti naują tekstinį failą „daina.txt“, į kurį įkelkite (nukopijuokite bei išsaugokite) dainos žodžius\*;
7. Taip pat Jūsų kuriamos dainos pavadinimo kataloge reikia sukurti naują katalogą „zodziai“, kuriame bus talpiname prodiuserių gaminami žodžiai;
8. Kataloge „zodziai“ sudėkite n prodiuserių kuriamų žodžių tekstinių failų, kuriuos pervardinkite nuo 1.txt iki n.txt, kur n – kuriančių prodiuserių skaičius.

\* PASTABA. Jei programos vykdymo metu naudosite prodiuserių kuriamų žodžių duomenų failų generaciją, atlikite tik 1-6 veiksmus;

## Vykdymo instrukcija

1. Pasileidžiame Command Prompt (Start -> Command Prompt);
2. Į komandinę eilutę įrašykite komandą, kuri nueina į nurodytą direktoriją. Šiuo atveju, reikia nurodyti katalogą „Daina“. Naudojame komandą:

cd {kelio pavadinimas iki katalogo dainą}

Pvz., jei katalogą sukūrėte savo darbalaukyje, tai nurodykite kelią taip:

cd C:\Users\{Jūsų vartotojo vardas}\Desktop\Daina

Mano atveju, komanda atrodo taip:

cd C:\Users\Mangirdas\Desktop\Daina.

1. Progamos paleidimui naudokite komandą:

java -jar ID.jar {programos parametrai}

Vietoj {programos parametrai} įveskite Jūsų norimus parametrus:

* Jei norite paleisti demonstracinę programos versiją įveskite tokius parametrus:

-d **{gijų skaičius}**, kur:

**{gijų skaičius}** – kiek gijų (prodiuserių) bus naudojama programos vykdymo metu (sveikasis skaičius, didesnis už 0).

**Komandos pavyzdys:** java -jar ID.jar -d 64

* Jei norite paleisti programos versiją su Jūsų daina, kurioje prodiuserių kuriami žodžiai yra generuojami, įveskite tokius parametrus:

-g “**{dainos pavadinimas}**“ **{gijų skaičius}** **{generavimo būdas}**, kur:

**{dainos pavadinimas}** – Jūsų norimos sukurti dainos pavadinimas;

**{gijų skaičius}** – kiek gijų (prodiuserių) bus naudojama programos vykdymo metu (sveikasis skaičius, didesnis už 0);

**{generavimo būdas}** – prodiuserių kuriamų žodžių generavimo būdas: jei 1 – žodžiai prodiuseriams skirstomi tolygiais dainos žodžių blokais, pradedant nuo pirmojo prodiuserio, o jei 2 - žodžiai prodiuseriams skirstomi po vieną, pradedant nuo pirmojo prodiuserio.

**Komandos pavyzdys:** java -jar ID.jar -g "Bohemian Rhapsody" 64 2

* Jei norite paleisti programos versiją su Jūsų daina, kurioje prodiuserių žodžių generacija nėra naudojama (juos nurodote Jūs), įveskite tokius parametrus:

-p “**{dainos pavadinimas}**“ **{gijų skaičius}**, kur:

**{dainos pavadinimas}** – Jūsų norimos sukurti dainos pavadinimas;

**{gijų skaičius}** – kiek gijų (prodiuserių) bus naudojama programos vykdymo metu (sveikasis skaičius, didesnis už 0);

**Komandos pavyzdys:** java -jar ID.jar -p "Bohemian Rhapsody" 64

# Vykdymo laiko kitimo tyrimas

Dėl JVM (JAVA virtualios mašinos) versijų skirtumų kompiuteryje (vykdant programą komandine eilute) bei programavimo aplinkoje (NetBeans IDE 8.1), dėl tikslesnių rezultatų programos vykdymo laiko kitimo tyrimui pasirinkta programą vykdyti NetBeans programavimo aplinkoje. Kompiuterio, su kuriuo buvo atliekamas laiko kitimo tyrimas, parametrai: 8 branduolių i7-3610QM 2.3 GHz procesorius, 12GB DDR3 RAM operatyviosios atminties, Windows 10 operacinė sistema.

Bus nagrinėjami tokie duomenų kiekiai: 5000, 10000, 20000, 40000, 80000 žodžių dainoje. Programa bus vykdoma su 1, 2, 8, 32, 64 bei 128 lygiagrečių procesų (gijų). Taip pat kiekvieną kartą bus nagrinėjami 2 atvejai: kai dainos žodžiai generacijos metu prodiuseriams bus skirstomi vienodo dydžio blokais, pradedant nuo pirmojo prodiuserio, bei kai dainos žodžiai kiekvienam prodiuseriui skirstomi po vieną, pradedant nuo pirmojo. Kiekvienas bandymas pakartotas po 5 kartus, bandymo rezultatu laikomas atliktų 5 bandymų vykdymo laikų vidurkis.

* **Žodžių skaičius dainoje – 5000:**

Programos vykdymo laiko tyrimas atliekamas su 5000 žodžių. Tyrimo rezultatai pateikti 9 lentelėje bei 1 ir 2 diagramoje.

lentelė. Programos vykdymo laikai pagal lygiagrečių procesų skaičių, kai žodžių dainoje - 5000

|  |  |  |
| --- | --- | --- |
| **Žodžių skaičius dainoje:** 5000 | | |
| Prodiuserių (gijų) skaičius | Programos vykdymo laikas (sekundėmis), kai žodžiai prodiuseriams paskirstyti blokais | Programos vykdymo laikas (sekundėmis), kai žodžiai prodiuseriams paskirstyti po vieną |
| 1 | 0,146 | 0,143 |
| 2 | 0,149 | 0,146 |
| 8 | 0,135 | 0,144 |
| 32 | 0,147 | 0,145 |
| 64 | 0,148 | 0,147 |
| 128 | 0,131 | 0,129 |

diagrama. Stulpelinė programos vykdymo laikų diagrama, kai žodžių skaičius – 5000

diagrama. Linijinis programos vykdymo laikų grafikas, kai žodžių skaičius – 5000

Kaip matome iš tyrimo rezultatų, programos vykdymo laikai labai panašūs, nepriklausomai nuo veikiančių lygiagrečių procesų skaičiaus. Galima pastebėti, kad greičiausiai programa veikė su 128 gijomis (pastebima, kad šiuo atveju programa veikė greičiau, kai duomenys prodiuseriams buvo paskirstyti po vieną). Programa veikė greičiausiai su 128 gijomis, lėčiausiai – su 2 gijomis.

* **Žodžių skaičius dainoje – 10000:**

Programos vykdymo laiko tyrimas atliekamas su 10000 žodžių. Tyrimo rezultatai pateikti 10 lentelėje bei 3 ir 4 diagramoje.

lentelė. Programos vykdymo laikai pagal lygiagrečių procesų skaičių, kai žodžių dainoje - 10000

|  |  |  |
| --- | --- | --- |
| **Žodžių skaičius dainoje:** 10000 | | |
| Prodiuserių (gijų) skaičius | Programos vykdymo laikas (sekundėmis), kai žodžiai prodiuseriams paskirstyti blokais | Programos vykdymo laikas (sekundėmis), kai žodžiai prodiuseriams paskirstyti po vieną |
| 1 | 0,495 | 0,535 |
| 2 | 0,489 | 0,479 |
| 8 | 0,522 | 0,476 |
| 32 | 0,331 | 0,330 |
| 64 | 0,323 | 0,309 |
| 128 | 0,359 | 0,357 |

diagrama. Stulpelinė programos vykdymo laikų diagrama, kai žodžių skaičius – 10000

diagrama. Linijinis programos vykdymo laikų grafikas, kai žodžių skaičius – 10000

Kaip matome iš tyrimo rezultatų, programos vykdymo laikų skirtumas, didėjant gijų skaičiui, tampa akivaizdus (programa veikia 0,15-0,2 sekundės greičiau, pvz., lyginant programos veikimą su 8 ir su 32 gijomis). Taip pat čia galima pastebėti tendenciją, kad pradedant nuo 2 gijų programos vykdymo, programa vykdoma greičiau, kai duomenys paskirstyti prodiuseriams po vieną žodį. Programa greičiausiai veikė su 64 gijomis, lėčiausiai – su 1 gija.

* **Žodžių skaičius dainoje – 20000:**

Programos vykdymo laiko tyrimas atliekamas su 20000 žodžių. Tyrimo rezultatai pateikti 11 lentelėje bei 5 ir 6 diagramoje.

lentelė. Programos vykdymo laikai pagal lygiagrečių procesų skaičių, kai žodžių dainoje - 20000

|  |  |  |
| --- | --- | --- |
| **Žodžių skaičius dainoje:** 20000 | | |
| Prodiuserių (gijų) skaičius | Programos vykdymo laikas (sekundėmis), kai žodžiai prodiuseriams paskirstyti blokais | Programos vykdymo laikas (sekundėmis), kai žodžiai prodiuseriams paskirstyti po vieną |
| 1 | 2,042 | 2,014 |
| 2 | 2,052 | 2,030 |
| 8 | 2,001 | 1,995 |
| 32 | 1,144 | 1,117 |
| 64 | 1,223 | 1,125 |
| 128 | 1,215 | 1,247 |

diagrama. Stulpelinė programos vykdymo laikų diagrama, kai žodžių skaičius – 20000

diagrama. Linijinis programos vykdymo laikų grafikas, kai žodžių skaičius – 20000

Kaip matome iš tyrimo rezultatų, programos vykdymo laikų skirtumas, didėjant gijų skaičiui, pvz., pereinant nuo 8 gijų prie 32 gijų, programos vykdymo greitis padidėja beveik 2 kartus. Tiesa, kai gijų skaičius siekia nuo 1 iki 8, vykdymo laikai panašūs (kaip ir 32 – 128 gijų), programa greičiausiai veikė su 32 gijomis, lėčiausiai – su 2 gijomis. Čia taip pat pastebima tendencija (1 – 64 gijų programos vykdyme), kad programa veikia greičiau, kai duomenys prodiuseriams paskirstyti po vieną.

* **Žodžių skaičius dainoje – 40000:**

Programos vykdymo laiko tyrimas atliekamas su 40000 žodžių. Tyrimo rezultatai pateikti 12 lentelėje bei 7 ir 8 diagramoje.

lentelė. Programos vykdymo laikai pagal lygiagrečių procesų skaičių, kai žodžių dainoje - 40000

|  |  |  |
| --- | --- | --- |
| **Žodžių skaičius dainoje:** 40000 | | |
| Prodiuserių (gijų) skaičius | Programos vykdymo laikas (sekundėmis), kai žodžiai prodiuseriams paskirstyti blokais | Programos vykdymo laikas (sekundėmis), kai žodžiai prodiuseriams paskirstyti po vieną |
| 1 | 10,849 | 11,094 |
| 2 | 10,714 | 10,834 |
| 8 | 5,511 | 4,786 |
| 32 | 4,317 | 4,383 |
| 64 | 4,344 | 4,335 |
| 128 | 4,380 | 4,403 |

diagrama. Stulpelinė programos vykdymo laikų diagrama, kai žodžių skaičius – 40000

diagrama. Linijinis programos vykdymo laikų grafikas, kai žodžių skaičius – 40000

Kaip matome iš tyrimo rezultatų, programos vykdymo laikų skirtumas, didėjant gijų skaičiui, taip pat didėja, t.y. programa veikia daugiau nei 2 kartus greičiau didėjant programoje vykdomų gijų skaičiui. Kaip matome, programa pradėjo ženkliai greičiau veikti jau tada, kai buvo vykdoma su 8 gijomis. Taip pat greičiausiai programa veikė tada, kai duomenys (žodžiai) prodiuseriams buvo paskirstyti po vieną. Greičiausiai programa veikė su 64 gijomis, lėčiausiai – su 1 gija.

* **Žodžių skaičius dainoje – 80000:**

Programos vykdymo laiko tyrimas atliekamas su 40000 žodžių. Tyrimo rezultatai pateikti 13 lentelėje bei 9 ir 10 diagramoje.

lentelė. Programos vykdymo laikai pagal lygiagrečių procesų skaičių, kai žodžių dainoje - 80000

|  |  |  |
| --- | --- | --- |
| **Žodžių skaičius dainoje:** 80000 | | |
| Prodiuserių (gijų) skaičius | Programos vykdymo laikas (sekundėmis), kai žodžiai prodiuseriams paskirstyti blokais | Programos vykdymo laikas (sekundėmis), kai žodžiai prodiuseriams paskirstyti po vieną |
| 1 | 21,135 | 21,859 |
| 2 | 20,884 | 21,421 |
| 8 | 20,081 | 20,095 |
| 32 | 20,357 | 20,305 |
| 64 | 20,668 | 20,878 |
| 128 | 20,808 | 21,330 |

diagrama. Stulpelinė programos vykdymo laikų diagrama, kai žodžių skaičius – 80000

diagrama. Linijinis programos vykdymo laikų grafikas, kai žodžių skaičius – 80000

Kaip matome iš tyrimo rezultatų, greičiausiai programa veikė, kai buvo vykdoma su 8 gijomis, lėčiausiai – su 1 gija. Šiuo atveju, programa veikė greičiau, kai duomenys prodiuseriams buvo paskirstyti blokais.

# Išvados

Individualaus darbo užduotis buvo atlikta naudojant JavaCSP paketo lygiagretaus programavimo priemonėmis. Iš bandymo rezultatų galima sakyti, kad šios priemonės programos realizavimui buvo tinkamos, nes dauguma atvejų, didinant procesų skaičių, t.y. programą vykdant lygiagrečiai daugiau gijų, greitaveika didėjo, programos vykdymas buvo greitesnis, prodiuserių kuriamų žodžių paiešką dainoje atliko sparčiau, todėl galima daryti bendrą individualaus darbo išvadą – JavaCSP lygiagretaus programavimo priemonės yra tinkamos lygiagrečiai vykdomų programų realizacijai, nes jos padeda pasiekti geresnius programos vykdymo laikus nei programą vykdant kaip vieną procesą, o programos veiksmus vykdant nuosekliai.

# Literatūra

1. <http://www.perl.lt/duomenys/veidrodziai/vgtu/Perl/regularexpr.htm>
2. <http://underpop.online.fr/j/java/algorithims-in-java-5/files/17fig10.gif>
3. <https://www.cs.kent.ac.uk/projects/ofa/jcsp/jcsp-1.1-rc4/jcsp-doc/org/jcsp/lang/Parallel.html>
4. Modulio „P170B328 Lygiagretusis programavimas“ paskaitos skaidrės MOODLE sistemoje: <https://moodle.ktu.edu/course/view.php?id=1977> (žiūrėta 2016-12-16)