VILNIAUS UNIVERSITETAS

MATEMATIKOS IR INFORMATIKOS FAKULTETAS

INFORMATIKOS KATEDRA

Dirbtinio intelekto laboratorinis darbas

**Atbulinis išvedimas produkcijų sistemoje**

**C++ programavimo kalba**

(Backward chaining)

Atliko: 4 kurso, 1 grupės studentas

Algirdas Martynas Vyčinas

Vilnius

2013

Turinys

[1 Algoritmo analizė 3](#_Toc374887290)

[1.1 Algoritmo aprašas 3](#_Toc374887291)

[1.2 Pagrindinio algoritmo (funkcijos) pseudokodas 3](#_Toc374887292)

[1.3 Algoritmo blokinė diagrama 4](#_Toc374887293)

[2 Programos veikimo pavyzdžiai 5](#_Toc374887294)

[2.1 Pirmas pavyzdys (3 produkcijos) 5](#_Toc374887295)

[2.1.1 Duomenys faile 5](#_Toc374887296)

[2.1.2 Programos išeities tekstas 5](#_Toc374887297)

[2.1.3 Semantinis grafas 5](#_Toc374887298)

[2.2 Antras pavyzdys (9 produkcijos, ciklas) 6](#_Toc374887299)

[2.2.1 Duomenys faile 6](#_Toc374887300)

[2.2.2 Programos išeities tekstas 6](#_Toc374887301)

[2.2.3 Semantinis grafas 7](#_Toc374887302)

[2.3 Trečias pavyzdys (9 produkcijos, ciklas) 7](#_Toc374887303)

[2.3.1 Duomenys faile 7](#_Toc374887304)

[2.3.2 Programos išeities tekstas 8](#_Toc374887305)

[2.3.3 Semantinis grafas 8](#_Toc374887306)

[2.4 Ketvirtas pavyzdys (5 produkcijos) 8](#_Toc374887307)

[2.4.1 Duomenys faile 8](#_Toc374887308)

[2.4.2 Programos išeities tekstas 9](#_Toc374887309)

[2.4.3 Semantinis grafas 9](#_Toc374887310)

[2.5 Penktas pavyzdys (5 produkcijos, Negnevitsky) 9](#_Toc374887311)

[2.5.1 Duomenys faile 9](#_Toc374887312)

[2.5.2 Programos išeities tekstas 10](#_Toc374887313)

[2.5.3 Semantinis grafas 10](#_Toc374887314)

[2.6 Šeštas pavyzdys (4 produkcijos, tikslas faktuose) 10](#_Toc374887315)

[2.6.1 Duomenys faile 10](#_Toc374887316)

[2.6.2 Programos išeities tekstas 11](#_Toc374887317)

[2.7 Septintas pavyzdys (8 produkcijos, nepasiekiamas tikslas) 11](#_Toc374887318)

[2.7.1 Duomenys faile 11](#_Toc374887319)

[2.7.2 Programos išeities tekstas 12](#_Toc374887320)

[2.7.3 Semantinis grafas 13](#_Toc374887321)

[3 Programos tekstas 14](#_Toc374887322)

[4 Šaltinių ir literatūros sąrašas 16](#_Toc374887323)

# Algoritmo analizė

## Algoritmo aprašas

Algoritmas yra formalizuotas, kad būtų lengviau suprantamas. Žemiau yra pateiktas algoritmo pseudokodas. Funkcija ***backward\_chaining*** priima 2 parametrus:

* ***goal*** – ieškomas tikslas;
* ***lygis*** – rekursijos lygis;

ir naudoja globalius kintamuosius:

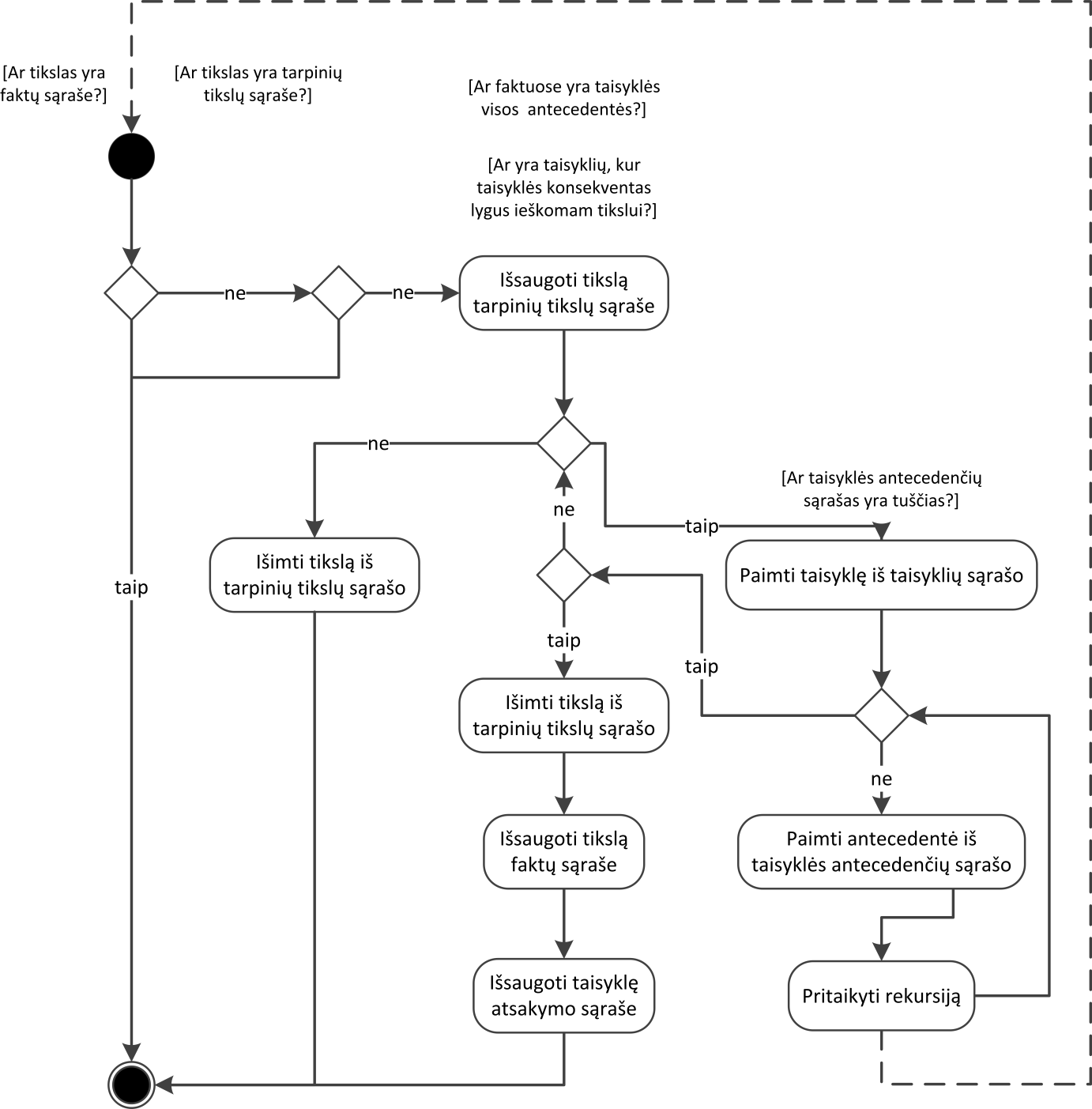
* ***taisykles*** – visų taisyklių vektorius,
* ***faktai*** – visų faktų aibė (vektorius),
* ***panaudotos\_taisykles*** *–* pritaikytų taisyklių vektorius.

Algoritmas pradedą darbą nuo tikslo ir bando rasti kelią iki pradinių duomenų, sudarant tarpinius tikslus. Jei tai pavyksta, gražinamas rezultatas kaip vektorius, jei ne – grąžinamas tuščias vektorius ir pranešama apie nesėkmę.

## Pagrindinio algoritmo (funkcijos) pseudokodas

|  |
| --- |
| backward\_chaining(goal, lygis)  1: **if** tikslas yra tarp faktų  2: **then** exit;  3: **if** tikslas nėra jau ieškotas  4: **then** **for** visos taisykles  5: **if** taisykle nepanaudota  6: **then** **if** taisykle veda į tikslą  7: **then** **begin**  8: reikalingos taisyklės += taisyklė  9: rasta produkcija = true  10: **end**  11: **end**  12: **if** neradome į tikslą vedančių taisyklų  13: **then** pasiekėme aklavietę  14: **else** ciklas  15: **for** reikalingos taisyklės  16: **for** taisyklės sąlygos  17: **if** sąlygos nėra tarp faktų  18: **then** backward\_chaining(sąlyga, lygis+1)  19: **end**  20: **if** taisyklės sąlygos yra patenkintos  21: **then** **return** rastų taisyklių sąrašas  22: **end**  23: **return** tuščias vektorius |

## Algoritmo blokinė diagrama



# Programos veikimo pavyzdžiai

## Pirmas pavyzdys (3 produkcijos)

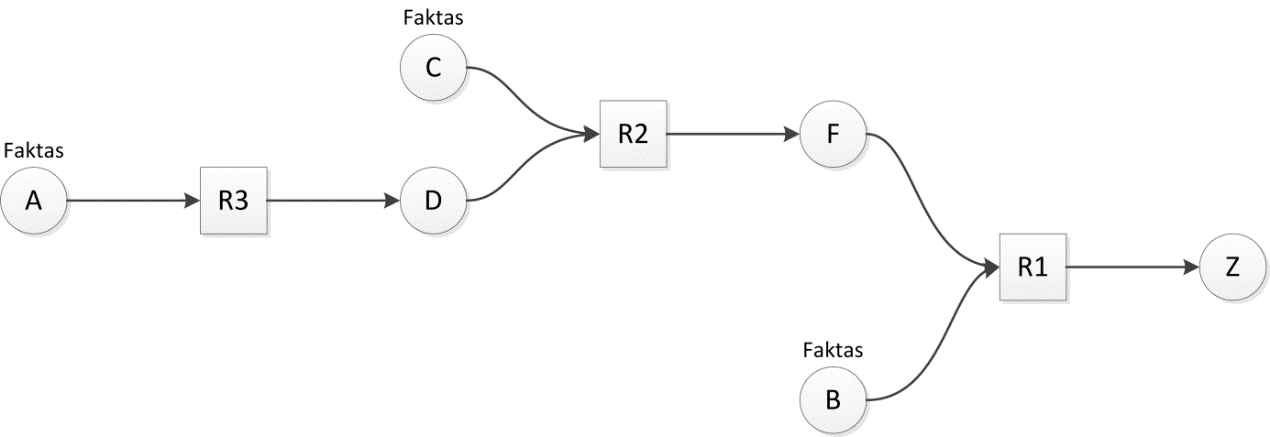
### Duomenys faile

|  |
| --- |
| 1) Taisyklės  Z F B // R1 : F, B -> Z  F C D // R2 : C, D -> F  D A // R3 : A -> D  2) Faktai  A B C  3) Tikslas  Z |

### Programos išeities tekstas

|  |
| --- |
| Programa pradeda darbą  1) Duomenys iš failo:  Taisyklės  R1: F, B -> Z  R2: C, D -> F  R3: A -> D  Pradiniai faktai  A, B, C  Tikslas  Z  2) Vykdymo žingsniai:  1 Tikslas Z. Randame R1: F, B -> Z. Nauji tikslai: F, B.  2 Tikslas F. Randame R2: C, D -> F. Nauji tikslai: C, D.  3 Tikslas C. Duotas faktas.  4 Tikslas D. Randame R3: A -> D. Nauji tikslai: A.  5 Tikslas A. Duotas faktas.  6 Tikslas D. Naujai gautas faktas.  7 Tikslas F. Naujai gautas faktas.  8 Tikslas B. Duotas faktas.  9 Tikslas Z. Naujai gautas faktas.  3) Rezultatas:  Tikslas Z pasiekiamas. Planas: {R3, R2, R1}  Programa baigia darbą |

### Semantinis grafas



## Antras pavyzdys (9 produkcijos, ciklas)

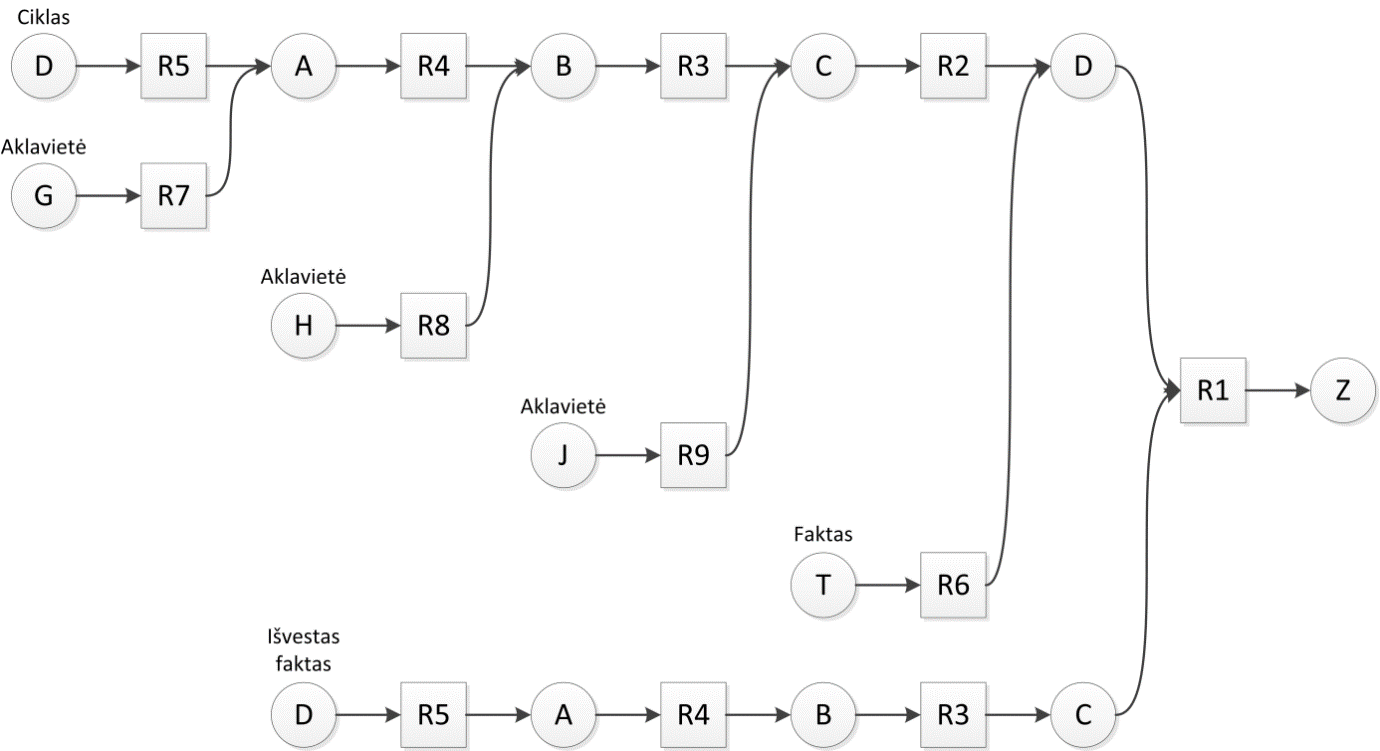
### Duomenys faile

|  |
| --- |
| 1) Taisyklės  Z D C // R1: D, C -> Z  D C // R2: C -> D  C B // R3: B -> C  B A // R4: A -> B  A D // R5: D -> A  D T // R6: T -> D  A G // R7: G -> A  B H // R8: H -> B  C J // R9: J -> C  2) Faktai  T  3) Tikslas  Z |

### Programos išeities tekstas

|  |
| --- |
| Programa pradeda darbą  1) Duomenys iš failo:  Taisyklės  R1: D, C -> Z  R2: C -> D  R3: B -> C  R4: A -> B  R5: D -> A  R6: T -> D  R7: G -> A  R8: H -> B  R9: J -> C  Pradiniai faktai  T  Tikslas  Z  2) Vykdymo žingsniai:  1 Tikslas Z. Randame R1: D, C -> Z. Nauji tikslai: D, C.  2 Tikslas D. Randame R2: C -> D. Nauji tikslai: C.  3 Tikslas C. Randame R3: B -> C. Nauji tikslai: B.  4 Tikslas B. Randame R4: A -> B. Nauji tikslai: A.  5 Tikslas A. Randame R5: D -> A. Nauji tikslai: D.  6 Tikslas D. Ciklas.  7 Tikslas A. Randame R7: G -> A. Nauji tikslai: G.  8 Tikslas G. Aklavietė (nėra taisyklių).  9 Tikslas A. Išvesti neįmanoma.  10 Tikslas B. Randame R8: H -> B. Nauji tikslai: H.  11 Tikslas H. Aklavietė (nėra taisyklių).  12 Tikslas B. Išvesti neįmanoma.  13 Tikslas C. Randame R9: J -> C. Nauji tikslai: J.  14 Tikslas J. Aklavietė (nėra taisyklių).  15 Tikslas C. Išvesti neįmanoma.  16 Tikslas D. Randame R6: T -> D. Nauji tikslai: T.  17 Tikslas T. Duotas faktas.  18 Tikslas D. Naujai gautas faktas.  19 Tikslas C. Randame R3: B -> C. Nauji tikslai: B.  20 Tikslas B. Randame R4: A -> B. Nauji tikslai: A.  21 Tikslas A. Randame R5: D -> A. Nauji tikslai: D.  22 Tikslas D. Duotas faktas.  23 Tikslas A. Naujai gautas faktas.  24 Tikslas B. Naujai gautas faktas.  25 Tikslas C. Naujai gautas faktas.  26 Tikslas Z. Naujai gautas faktas.  3) Rezultatas:  Tikslas Z pasiekiamas. Planas: {R6, R5, R4, R3, R1}  Programa baigia darbą |

### Semantinis grafas



## Trečias pavyzdys (9 produkcijos, ciklas)

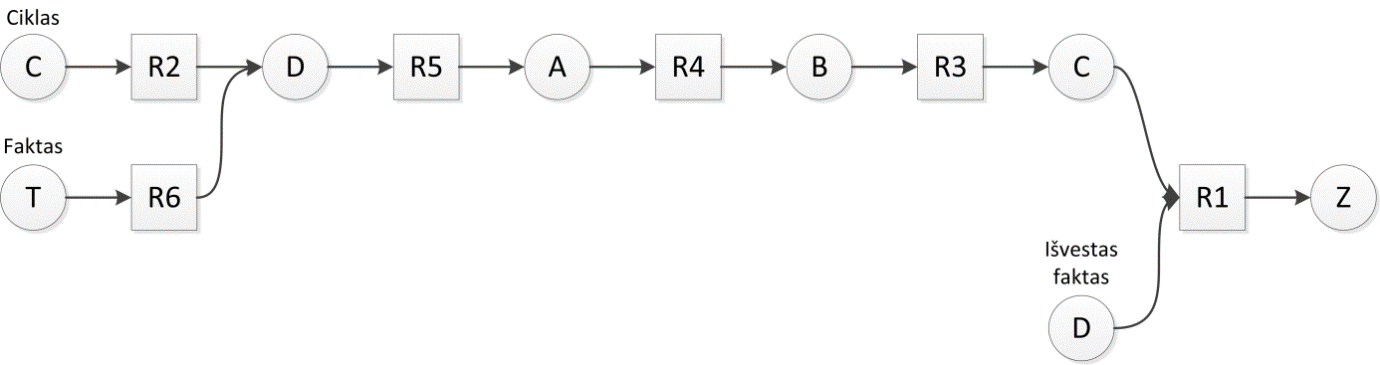
### Duomenys faile

|  |
| --- |
| 1) Taisyklės  Z C D // R1: C, D -> Z  D C // R2: C -> D  C B // R3: B -> C  B A // R4: A -> B  A D // R5: D -> A  D T // R6: T -> D  A G // R7: G -> A  B H // R8: H -> B  C J // R9: J -> C  2) Faktai  T  3) Tikslas  Z |

### Programos išeities tekstas

|  |
| --- |
| Programa pradeda darbą  1) Duomenys iš failo:  Taisyklės  R1: C, D -> Z  R2: C -> D  R3: B -> C  R4: A -> B  R5: D -> A  R6: T -> D  R7: G -> A  R8: H -> B  R9: J -> C  Pradiniai faktai  T  Tikslas  Z  2) Vykdymo žingsniai:  1 Tikslas Z. Randame R1: C, D -> Z. Nauji tikslai: C, D.  2 Tikslas C. Randame R3: B -> C. Nauji tikslai: B.  3 Tikslas B. Randame R4: A -> B. Nauji tikslai: A.  4 Tikslas A. Randame R5: D -> A. Nauji tikslai: D.  5 Tikslas D. Randame R2: C -> D. Nauji tikslai: C.  6 Tikslas C. Ciklas.  7 Tikslas D. Randame R6: T -> D. Nauji tikslai: T.  8 Tikslas T. Duotas faktas.  9 Tikslas D. Naujai gautas faktas.  10 Tikslas A. Naujai gautas faktas.  11 Tikslas B. Naujai gautas faktas.  12 Tikslas C. Naujai gautas faktas.  13 Tikslas D. Duotas faktas.  14 Tikslas Z. Naujai gautas faktas.  3) Rezultatas:  Tikslas Z pasiekiamas. Planas: {R6, R5, R4, R3, R1}  Programa baigia darbą |

### Semantinis grafas



## Ketvirtas pavyzdys (5 produkcijos)

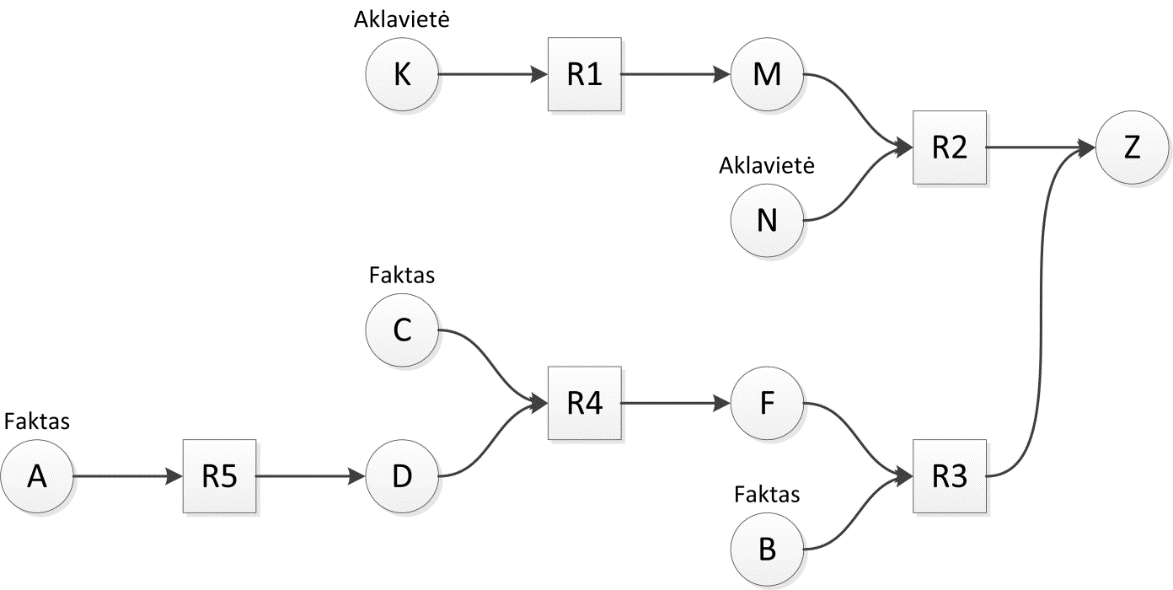
### Duomenys faile

|  |
| --- |
| 1) Taisyklės  M K // R1: K -> M  Z M N // R2: M, N -> Z  Z F B // R3: F, B -> Z  F C D // R4: C, D -> F  D A // R5: A -> D  2) Faktai  A B C  3) Tikslas  Z |

### Programos išeities tekstas

|  |
| --- |
| Programa pradeda darbą  1) Duomenys iš failo:  Taisyklės  R1: K -> M  R2: M, N -> Z  R3: F, B -> Z  R4: C, D -> F  R5: A -> D  Pradiniai faktai  A, B, C  Tikslas  Z  2) Vykdymo žingsniai:  1 Tikslas Z. Randame R2: M, N -> Z. Nauji tikslai: M, N.  2 Tikslas M. Randame R1: K -> M. Nauji tikslai: K.  3 Tikslas K. Aklavietė (nėra taisyklių).  4 Tikslas M. Išvesti neįmanoma.  5 Tikslas Z. Randame R3: F, B -> Z. Nauji tikslai: F, B.  6 Tikslas F. Randame R4: C, D -> F. Nauji tikslai: C, D.  7 Tikslas C. Duotas faktas.  8 Tikslas D. Randame R5: A -> D. Nauji tikslai: A.  9 Tikslas A. Duotas faktas.  10 Tikslas D. Naujai gautas faktas.  11 Tikslas F. Naujai gautas faktas.  12 Tikslas B. Duotas faktas.  13 Tikslas Z. Naujai gautas faktas.  3) Rezultatas:  Tikslas Z pasiekiamas. Planas: {R5, R4, R3}  Programa baigia darbą |

### Semantinis grafas



## Penktas pavyzdys (5 produkcijos, Negnevitsky)

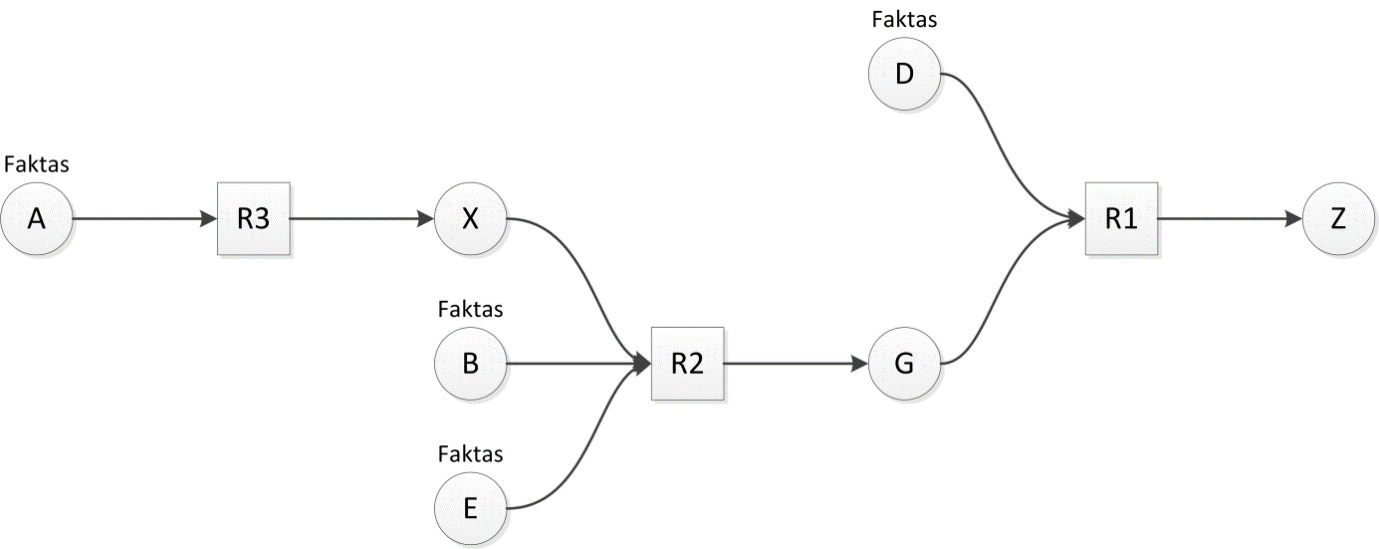
### Duomenys faile

|  |
| --- |
| 1) Taisyklės  Z D Y // R1: D, Y -> Z  Y X B E // R2: X, B, E -> Y  X A // R3: A -> X  L C // R4: C -> L  N L M // R5: L, M -> N  2) Faktai  A B D E  3) Tikslas  Z |

### Programos išeities tekstas

|  |
| --- |
| Programa pradeda darbą  1) Duomenys iš failo:  Taisyklės  R1: D, Y -> Z  R2: X, B, E -> Y  R3: A -> X  R4: C -> L  R5: L, M -> N  Pradiniai faktai  A, B, D, E  Tikslas  Z  2) Vykdymo žingsniai:  1 Tikslas Z. Randame R1: D, Y -> Z. Nauji tikslai: D, Y.  2 Tikslas D. Duotas faktas.  3 Tikslas Y. Randame R2: X, B, E -> Y. Nauji tikslai: X, B, E.  4 Tikslas X. Randame R3: A -> X. Nauji tikslai: A.  5 Tikslas A. Duotas faktas.  6 Tikslas X. Naujai gautas faktas.  7 Tikslas B. Duotas faktas.  8 Tikslas E. Duotas faktas.  9 Tikslas Y. Naujai gautas faktas.  10 Tikslas Z. Naujai gautas faktas.  3) Rezultatas:  Tikslas Z pasiekiamas. Planas: {R3, R2, R1}  Programa baigia darbą |

### Semantinis grafas



## Šeštas pavyzdys (4 produkcijos, tikslas faktuose)

### Duomenys faile

|  |
| --- |
| 1) Taisyklės  M N // R1: N -> M  D A // R2: A -> D  K D // R3: D -> K  A B // R4: B -> A  2) Faktai  A B C Z  3) Tikslas  Z |

### Programos išeities tekstas

|  |
| --- |
| Programa pradeda darbą  1) Duomenys iš failo:  Taisyklės  R1: N -> M  R2: A -> D  R3: D -> K  R4: B -> A  Pradiniai faktai  A, B, C, Z  Tikslas  Z  2) Vykdymo žingsniai:  1 Tikslas Z. Duotas faktas.  3) Rezultatas:  Tikslas Z pasiekiamas. Planas: {}  Programa baigia darbą |

## Septintas pavyzdys (8 produkcijos, nepasiekiamas tikslas)

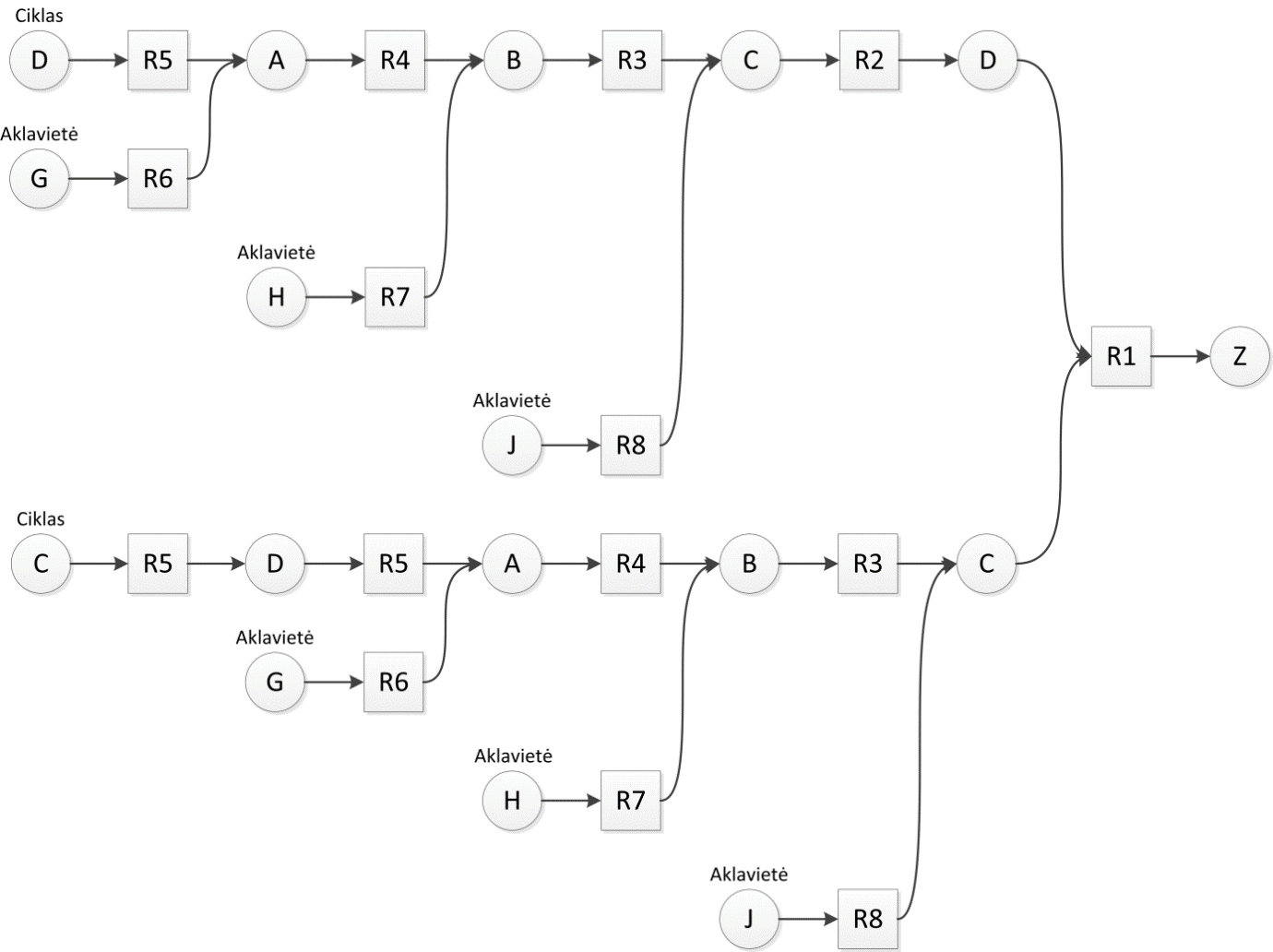
### Duomenys faile

|  |
| --- |
| 1) Taisyklės  Z D C // R1: D, C -> Z  D C // R2: C -> D  C B // R3: B -> C  B A // R4: A -> B  A D // R5: D -> A  A G // R6: G -> A  B H // R7: H -> B  C J // R8: J -> C  2) Faktai  T  3) Tikslas  Z |

### Programos išeities tekstas

|  |
| --- |
| Programa pradeda darbą  1) Duomenys iš failo:  Taisyklės  R1: D, C -> Z  R2: C -> D  R3: B -> C  R4: A -> B  R5: D -> A  R6: G -> A  R7: H -> B  R8: J -> C  Pradiniai faktai  T  Tikslas  Z  2) Vykdymo žingsniai:  1 Tikslas Z. Randame R1: D, C -> Z. Nauji tikslai: D, C.  2 Tikslas D. Randame R2: C -> D. Nauji tikslai: C.  3 Tikslas C. Randame R3: B -> C. Nauji tikslai: B.  4 Tikslas B. Randame R4: A -> B. Nauji tikslai: A.  5 Tikslas A. Randame R5: D -> A. Nauji tikslai: D.  6 Tikslas D. Ciklas.  7 Tikslas A. Randame R6: G -> A. Nauji tikslai: G.  8 Tikslas G. Aklavietė (nėra taisyklių).  9 Tikslas A. Išvesti neįmanoma.  10 Tikslas B. Randame R7: H -> B. Nauji tikslai: H.  11 Tikslas H. Aklavietė (nėra taisyklių).  12 Tikslas B. Išvesti neįmanoma.  13 Tikslas C. Randame R8: J -> C. Nauji tikslai: J.  14 Tikslas J. Aklavietė (nėra taisyklių).  15 Tikslas C. Išvesti neįmanoma.  16 Tikslas D. Išvesti neįmanoma.  17 Tikslas Z. Išvesti neįmanoma.  3) Rezultatas:  Tikslas nepasiekiamas.  Programa baigia darbą |

### Semantinis grafas



# Programos tekstas

|  |
| --- |
| 001 vector**<**char**>** faktai**;** // visų faktų vektorius  002 string tikslas**;** // pradinis tikslas  003 vector**<**string**>** taisykles**;** // visų taisyklių vektorius  004 vector**<**int**>** panaudotos\_taisykles**;** // panaudotų taisyklių vektorius  005 vector**<**char**>** ieskoti\_tikslai**;** // ieškotų tikslų vektorius  006 char tmp **=** ' '**;** // laikinas kintamasis saugoti tikslui tarp rekursijos lygių  007 vector**<**int**>** kelias**;** // vektorius, kuriame kaupiamas atsakymas  008 bool need\_to\_print **=** **true;** // flag'as, kuris žymi ar reikią spausdintį tikslą  009 int zingsniu\_skaicius **=** 0**;** // žingsnių skaitliukas  010 int zingsnis\_kartojas **=** 0**;** // flag'as, kuris žymi ar paskutinis spausdintas žingsnis kartojas  011 char buves\_tikslas**;** // laikinas kintamasis, saugoti buvusiam tikslui  012  013 // Pagrindinė rekursyvi algoritmo funkcija  014 vector**<**int**>** backward\_chaining**(**char goal**,** int lygis**){**  015 bool rasta\_produkcija **=** **false;** // flag'as, žymi ar rasta bent 1 tinkama produkcija  016 vector**<**int**>** reikalingos\_taisykles**;** // vektorius, kuriame bus saugomos tikslui  017 // pasiekti reikalingos taisyklės  018  019 zingsniu\_skaicius**++;**  020 print\_zingsniai**(**zingsniu\_skaicius**,** lygis**);**  021 cout **<<** "Tikslas "**<<** goal **<<**". "**;**  022 // 1: Tikriname ar tisklas nėra tarp duotųjų faktų  023 **if** **(**ar\_tikslas\_tarp\_faktu**(**goal**)** **){**  024 // 2: Jei yra, tuomet išeiti  025 exit**(**0**);**  026 **}**  027 need\_to\_print **=** **false;**  028 // 3: Jei tikslas nėra jau ieškotas  029 **if(**std**::**find**(**ieskoti\_tikslai**.**begin**(),** ieskoti\_tikslai**.**end**(),**  030 goal**)** **==** ieskoti\_tikslai**.**end**()){**  031 rasta\_produkcija **=** **false;**  032 // 4: Einame per visas taisykles:  033 **for** **(**int i **=** 0**;** i **<** taisykles**.**size**()** **;** i**++** **)** **{**  034 // 5: Jei taisyklė yra nepanaudota:  035 **if** **(** std**::**find**(**panaudotos\_taisykles**.**begin**(),** // ar produkcija yra tarp panaudotu produkciju:  036 panaudotos\_taisykles**.**end**(),** i**)==**panaudotos\_taisykles**.**end**()** **){**  037 // 6: Jei taisyklė veda į tikslą  038 **if** **(**taisykles**[**i**][**0**]** **==** goal**)**  039 // 7: Tuomet pradėti  040 **{**  041 // 8: Pasižymime ją kaip reikalingą:  042 reikalingos\_taisykles**.**push\_back**(**i**);**  043 // 9: Pasižymime, kad radome į tikslą vedančią taisyklę  044 rasta\_produkcija **=** **true;**  045 // 10: Baigti  046 **}**  047 **}**  048 // 11: Baigiame eiti per taisykles  049 **}**  050 // 12: Jei nė viena taisyklė nerasta,  051 **if** **(!**rasta\_produkcija**)** **{**  052 // 13: Vadinasi pasiekėme aklavietę:  053 cout **<<** "Aklavietė (nėra taisyklių).\n"**;**  054 tmp **=** goal**;**  055 **}**  056 **}**  057 // 14: Jei tikslas jau buvo ieškotas, vadinasi priėjome ciklą:  058 **else** **{**  059 cout **<<** "Ciklas.\n"**;**  060 tmp **=** goal**;**  061 **}**  062 // 15: Einame per tikslui pasiekti reikalingas taisykles  063 **for** **(**int rule **=** 0**;** rule **<** reikalingos\_taisykles**.**size**();** **++**rule**)** **{**  064 int taisykle **=** reikalingos\_taisykles**[**rule**];**  065 // Pazymime, kad taisykles isvados jau ieskojome  066 ieskoti\_tikslai**.**push\_back**(**taisykles**[**taisykle**][**0**]);**  067 // Prie kelio pridedame sia taisykle  068 panaudotos\_taisykles**.**push\_back**(**taisykle**);**  069 vector **<**int**>** rastu\_taisykliu\_list**;**  070 bool patenkintos\_salygos **=** **true;**  071 **if** **(**need\_to\_print**){**  072 zingsniu\_skaicius**++;**  073 print\_zingsniai**(**zingsniu\_skaicius**,** lygis**);**  074 cout **<<** "Tikslas "**<<** goal **<<**". "**;**  075 **}**  076 cout **<<** "Randame R" **<<** taisykle**+**1 **<<** print\_taisykle**(**taisykle**)** **<<** ". "**;**  077 cout **<<** "Nauji tikslai: "**;**  078 salygos**(**taisykles**[**taisykle**]);**  079 cout **<<** ".\n"**;**  080 kelias**.**push\_back**(**taisykle**);**  081 bool ivykdoma **=** **true;**  082 // 16: Einame per taisyklės salygas:  083 **for** **(**int i **=** 1**;** i **<** taisykles**[**taisykle**].**size**()** **&&** ivykdoma**;** **++**i**)** **{**  084 char salyga **=** taisykles**[**taisykle**].**at**(**i**);**  085 // 17: Jei taisyklės sąlygos nėra tarp faktų:  086 **if** **(**std**::**find**(**faktai**.**begin**(),** faktai**.**end**(),**  087 salyga**)==**faktai**.**end**())** **{**  088 vector **<**int**>** kelias\_temp**;**  089 patenkintos\_salygos **=** **false;**  090 // 18: Tuomet tą salygą pridedame kaip tarpinį tikslą:  091 kelias\_temp **=** backward\_chaining**(**salyga**,** lygis**+**1**);**  092 **if** **(**kelias\_temp**.**size**()** **==** 0**)**  093 ivykdoma **=** **false;**  094 **for** **(**int j **=** 0**;** j **<** kelias\_temp**.**size**();** **++**j**)** **{**  095 rastu\_taisykliu\_list**.**push\_back**(**kelias\_temp**[**j**]);**  096 **}**  097 **}**  098 **if** **(**tmp **==** ' '**){**  099 zingsniu\_skaicius**++;**  100 tmp **=** salyga**;**  101 print\_zingsniai**(**zingsniu\_skaicius**,** lygis**+**1**);**  102 cout **<<** "Tikslas "**<<** salyga**;**  103 cout **<<** ". Duotas faktas.\n"**;**  104 **}**  105 **else** **{**  106 **if(**tmp **!=** salyga**){**  107 tmp **=** salyga**;**  108 zingsniu\_skaicius**++;**  109 print\_zingsniai**(**zingsniu\_skaicius**,** lygis**+**1**);**  110 cout **<<** "Tikslas "**<<** salyga**;**  111 cout **<<**". Duotas faktas.\n"**;**  112 **}**  113 **else** **{**  114 tmp **=** ' '**;**  115 **}**  116 **}**  117 // 19: Baigiame eiti per taisykles  118 **}**  119 // 20: Jei taisyklės visos sąlygos yra patenkintos:  120 **if** **(**salygos\_patenkintos**(**taisykle**))** **{**  121 char isvada **=** taisykles**[**taisykle**][**0**];**  122 faktai**.**push\_back**(**isvada**);**  123 rastu\_taisykliu\_list**.**push\_back**(**taisykle**);**  124 zingsniu\_skaicius**++;**  125 print\_zingsniai**(**zingsniu\_skaicius**,** lygis**);**  126 cout **<<** "Tikslas "**<<** taisykles**[**taisykle**][**0**];**  127 cout **<<** ". Naujai gautas faktas. \n"**;**  128 tmp **=** taisykles**[**taisykle**][**0**];**  129 // 21: Tuomet grąžiname rastų taisyklių sąrašą (vektorių):  130 **return** rastu\_taisykliu\_list**;**  131 **}**  132 **else** **{**  133 **if** **(**zingsnis\_kartojas **==** zingsniu\_skaicius**){**  134 zingsniu\_skaicius**++;**  135 print\_zingsniai**(**zingsniu\_skaicius**,** lygis**);**  136 cout **<<** "Tikslas "**<<** buves\_tikslas**;**  137 cout **<<** ". Išvesti neįmanoma.\n"**;**  138 zingsnis\_kartojas **=** 0**;**  139 **}**  140 **else** **{**  141 zingsnis\_kartojas **=** zingsniu\_skaicius**;**  142 buves\_tikslas **=** goal**;**  143 **}**  144 need\_to\_print **=** **true;**  145 tmp **=** taisykles**[**taisykle**][**0**];**  146 **}**  147 // Taisyklė pašalinama iš panaudotų produkcijų sąrašo:  148 vector**<**int**>::**iterator position **=** std**::**find**(**panaudotos\_taisykles**.**begin**(),** panaudotos\_taisykles**.**end**(),** taisykle**);**  149 **if** **(**position **!=** panaudotos\_taisykles**.**end**())**  150 panaudotos\_taisykles**.**erase**(**position**);**  151 // Taisyklės išvada pašalinama iš ieškotų tikslų sąrašo:  152 vector**<**char**>::**iterator position2 **=** std**::**find**(**ieskoti\_tikslai**.**begin**(),** ieskoti\_tikslai**.**end**(),** taisykles**[**taisykle**][**0**]);**  153 **if** **(**position2 **!=** ieskoti\_tikslai**.**end**())**  154 ieskoti\_tikslai**.**erase**(**position2**);**  155 // 22: Baigiame eiti per reikalingas taisykles  156 **}**  157 **if** **(**need\_to\_print**){**  158 zingsniu\_skaicius**++;**  159 print\_zingsniai**(**zingsniu\_skaicius**,** lygis**);**  160 cout **<<** "Tikslas "**<<** goal **<<**". "**;**  161 cout **<<** "Išvesti neįmanoma.\n"**;**  162 **}**  163 vector **<**int**>** empty**;**  164 // 23: Gražiname tuščią vektorių:  165 **return** empty**;**  166 **}** |

# Šaltinių ir literatūros sąrašas

[Čyr12] Vytautas Čyras. Artificial intelligence, 2012. Prieiga per internetą: http://uosis.mif.vu.lt/~cyras/AI/ai-cyras.pdf

[Til11] Aidas. Tilmantas. Tiesioginis ir atbulinis išvedimas produkcijų   
sistemoje, 2011. Prieiga per internetą: http://uosis.mif.vu.lt/~cyras/AI/201112-AI-Tilmantas-FC-BC-Java.zip

[Neg05] Michael Negnevitsky: Artificial Intelligence: A Guide To Intelligent Systems, 2005.