# Įvadas į PKT

**Kokie yra privalumai, kai kalbą įgyvendiname grynuoju interpretatoriumi?**

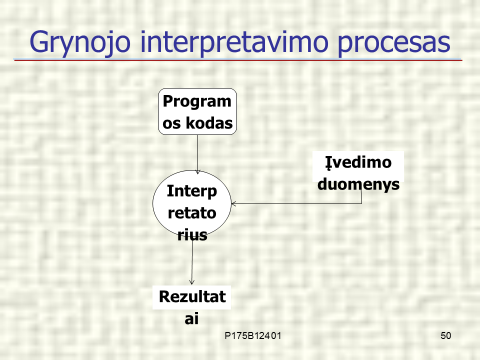
Programas interpretuoja kita programa, vadinama interpretatoriumi

Jokio transliavimo – tai akivaizdi virtuali mašina

Lengvesnis programų įgyvendinimas (sprendimo laiko klaidos gali būti lengvai aptiktos ir parodytos)

**Koks yra svarbiausias grynojo interpretatoriaus trūkumas?**

* x Lėtesnis vykdymas (nuo 10 iki 100 kartų lėčiau, nei kompiliuotos programos)
* Dažnai reikalauja daugiau erdvės
* Dabar retai naudojamas aukšto lygmens kalboms
* Sakinių dekodavimas – siauroji vieta, nes kiekvieną sykį vykdymo metu reikia dekoduoti tą patį sakinį, taip pat reikia ir simbolių lentelės

****

**Koks yra svarbiausias von Neumann architektūros trūkumas?**

* Greitis tarp atminties ir jo procesoriaus asprendžia kompiuterio greitį. Programos komandos galėtų būti vykdomos dažnai greičiau nei šis ryšio greitis. Tai siauroji vieta, pirminis kompiuterių greičio ribojimo veiksnys.

**Kada kompiliatoriuje naudojama simbolių lentelė?**

* Simbolių lentelė kompiliatoriui reikalinga saugoti kintamuosius su jų parametrais vienoje struktūrizuotoje vietoje.
* Kai tik kompiliatoriui prireikia kokio nors vardo, jis ieško jo simbolių lentelėje.

**Ką reiškia, kad programa yra patikima?**

* Programa patikima, jei dirba pagal specifikaciją prie bet kokių sąlygų
* Tipų tikrinimas
  + Tipų klaidų testavimas, ankstyvas aptikimas, vykdymo metu tipų klaidų tikrinimas brangus
* Išimčių valdymas
  + Perimti vykdymo klaidas ir imtis koregavimo priemonių
* Sinonimai
  + Du ar daugiau būdų pasiekti tą pačią atminties ląstelę
* Skaitomumas ir rašomumas
  + Jei nėra natūralių būdų, tenka naudoti nenatūralius būdus – mažėja patikimumas

**Kas turėjo didžiausią įtaką programavimo kalbų projektams per prabėgusius 50 metų?**

* Didžiausią įtaką kalbų projektams turi kompiuterių architektūra ir programavimo metodologijos
* 1950 ir 1960 pradžia: paprastos programos – neefektyvi mašina
* 1960 pabaiga: žmonių efektyvumas tampa svarbus; skaitomumas, geresnės valdymo struktūros
  + Struktūrinis programavimas
  + Iš viršaus žemyn projektavimas, žingsninis detalizavimas
* 1970 pabaiga: procesus orientuotos į duomenis orientuotos
  + Duomenų abstrakcija
* 1980 vidurys: objektinis programavimas
  + Duomenų abstrakcija + paveldėjimas + polimorphizmas
* Lygiagretumas

**Kaip vadinima kalbų, kurių struktūra yra tiesioiai įtakota von Neumann architektūros, kategorija?**

* Liepiamoji (kintamieji modeliuoja atminties ląsteles, priskyrimo sakiniai modeliuoja duomenų perkėlimą, efektyvios iteracijos, nes komandos šalia atminties ląstelės)

**Kokia programavimo kalbos konstrukcija suteikia proceso abstrakciją?**

* Abstrakcijos galimybės
  + Galimybė apibrėžti ir naudoti sudėtingas struktūras, ignoruojant jų realizavimo detales. Ypač svarbi abstrakcija šiandien
  + Proceso abstrakcija (**paprogramės** - rikiavimas) ir duomenų abstrakcija (dvejetainiai medžiai Fortran ir C++)

**Kas yra sinonimai programavime?**

* + Du ar daugiau būdų pasiekti tą pačią atminties ląstelę

**Pateikite bent vieną ortogonalumo pažeidimo atvejį C kalbos projekte.**

* + Ortogonalumas – tai galimybė gauti naujas kalbos konstrukcijas visais duotais būdais kombinuojant turimas konstrukcijas. Kuo didesnis kalbos ortogonalumas, tuo mažiau išimčių joje. Kai kalba ortogonali, tai iš nedidelio elementarių (pradinių) konstrukcijų su nedaugeliu kombinacijų galime gauti visas valdymo ir duomenų struktūras. Kiekviena kombinacija yra teisėta ir prasminga. 12 Labiausiai ortogonali kalba yra Algolas-68. Joje, pavyzdžiui nėra skirtumo tarp reiškinio ir sakinio. Jų funkcijas atlieka viena ir ta pati konstrukcija, turinti reikšmę ir tipą (kaip sakinys) ir galinti atlikti priskyrimo veiksmą arba kreiptis į procedūrą (kaip sakinys)
  + We can convert from integer to float by simply assigning a float to an integer, but not vice versa. (not a question of ability to do – generality, but of the way it is done)
  + Arrays are pass by “what appears to be” reference while integers are pass by value.
  + A switch statement works with integers, characters, or enumerated types, but not doubles or Strings.
  + Structures (but not arrays) may be returned from a function. An array can be returned if it is inside a structure.

# Istorija

**Kokia masyvo struktūra yra įtraukta į C# bet ne į C, C++ arba Java?**

* Rectangular array (jagged array)

**Kuo Ruby aritmenių operatorių charakteriskos išskiria iš kitų programavimo kalbų?**

* Populiariausi operatoriai yra įgyvendinti kaip metodai, kuriuos vartotojas gali pakeisti

**Koks C kalbos switch sakinio trūkumas ištaisytas C# kalboje?**

* Sudėtinga struktūra switch viduje

**Kaip pseudokodas buvo rašomas 1950-1960 metais?**

* Naudojant mašininį kodą

**Kokias dvi kalbas buvo norima pakeisti kuriant PL/I kalbą?**

* Fortran, Cobol

**Kodėl BASIC buvo svarbi kalba ankstyvais 1980 metais?**

* Pirmoji plačiai naudojama kalba su laiko pasidalinimu

**Kokia pirminė aplikacija , kuriai buvo sukurta Ada?**

* Įterptinė sistema

**Kokia Java charakteriska yra ryškiausia JavaScript kalboje?**

* Sintaksė

**Kokios bendros charakterikos tarp Ruby ir Smalltalk?**

* Abi pilnai objektinės (pure object-oriented)

**Kokia duomenų struktūra yra Lua kalboje?**

* Viskas vienoje duomenų struktūroje - lentelėje

**APL –** aparatūros aprašymo kalba IBM(daug operatorių skaliarams ir masyvams)

**SNOBOL –** eilučių manipuliavimo kalba(galingi operatoriai eilučių paieškai)

SIMULA 67 – Paprogramių rūšis – koprogramė, klasės, objektai, paveldimumas.

Koprogramė – paprogramių apibendrimas su daugybe įėjimo taškų, stabdymu ir vėlesniu tęsimu

Mašinos trūkumai – nėra indeksavimo ir realiųjų skaičių

# Sintaksė ir semantika

**Papasakokite kalbos atpažiniklio veikimą**

Atpažinimo įtaisas skaito įvedimo eilutes, sudarytas iš kalbos alfabeto, ir nusprendžia, ar įvedimo eilutė priklauso kalbai. Veikia kaip filtras

**Kas yra bekontekstė gramatika ir koks jos tikslas?**

Bekontekstė gramatika yra tam tikras formalios gramatikos tipas.

Kalbų generatoriai, skirti aprašyti natūralių kalbų sintaksę.  
Jos veikimo principas -> Paversti programos eilutes į suprantamą formatą.

Pvz A -> **α,** arba A -> **Ω**

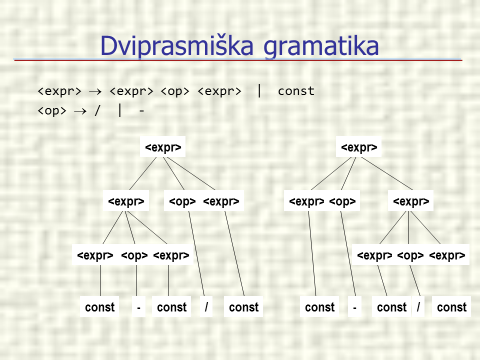
Šios taisyklės gali būti 1 -> 1, 1 -> \*, 1 -> 0

**Kada buvo sukurta BNF (Backus-Naur forma, John Backus) ir koks buvo jos tikslas?**

1959 metais.

Jos tikslas yra formaliai ir suprantamai aprašyti kalbos sintaksę, kai reikalingas konkretus jos apibrėžimas. Jis naudojamas vadovėliuose, manual‘uose ir panašiai. Atsirado, kadangi nebuvo standarto kaip aprašyti programavimo kalbos sintaksę ir gramatiką.

Tikslas – aprašyti Algol 58 sintaksę. Pirmą kartą panaudota, aprašant Algol 60 sintaksę.

**Kada bekontekstė gramatika yra dviprasmiška?**

* Gramatika yra dviprasmiška tada ir tik tada, jei ji generuoja sakinio formą, kuriai galima sudaryti keletą skirtingų gramatinių medžių

**Kas yra atributų gramatika?**

Tai tarsi papildymas (CFG) Context-Free Grammar, atnešantis tam tikrą semantiką į gramatinį medį. Atributai, atributų skaičiavimas, predikatų funkcijos.

Atributų gramatika yra bekontekstė gramatika G = (S, N, T, P) su tokiais papildymais:

* Kiekvienam gramatikos simboliui *x* yra aibė atributo reikšmių *A(x) (S(x) ir I(x) – nesusikertančios aibės)*
* Kiekviena taisyklė turi aibę funkcijų, kurios skaičiuoja tam tikrus neterminalo atributus taisyklėje
* Kiekviena taisyklė turi (gali būti tuščia) aibę predikatų, kad patikrintų atributų suderinamumą

**Kokius atributų tipus žinote?**

* Du tipai: sintezuojami ir paveldėti

**Kas yra medžio dekoravimas?**

* Atributų įvertinimo procesas (The process of evaluating attributes)

**Išvardinkite semantikos aprašymo metodus.**

Veiklos, žymėjimo, aksiomų

**Pristatykite būdą, kurį naudoja aksiomų semantika programų teisingumo įrodymui**

Einama atgal per programą atgal link pirmojo sakinio. Jei pirmojo sakinio išankstinė sąlyga yra tokia pati, kaip nurodyta programos specifikacijoje, tai programa teisinga

# LS Analizė

**Kokios yra 3 priežastys, kodėl sintaksės analizatoriai remiasi BNF?**

* Aiškus ir glaustas sintaksės aprašas tiek žmonėms, tiek programoms
* Analizatorius gali būti tiesiogiai bazuojamas BNF
* Analizatorius, kurie bazuojami BNF, lengva prižiūrėti dėl jų moduliškumo

**Paaiškinkite 3 priežastis, kodėl leksinė analizė yra atskirta nuo sintaksinės analizės?**

* Paprastumas – leksinei analizei naudojami mažiau sudėtingi metodai. Jų atskyrimas supaprastina analizatorių
* Efektyvumas – atskyrimas leidžia optimizuoti leksinį analizatorių
* Mobilumas – leksinio analizatoriaus dalys gali būti nepernešamos, bet sintaksės analizatorius visados pernešamas

**Kas yra leksinis analizatorius?**

* Jo tikslas yra transformuoti programos atvaizdavimą. Jis surenka simbolius į logines grupes ir priskiria vidinius kodus pagal grupių struktūrą. Loginės grupės vadinamos leksemomis ir šių grupių vidiniai kodai yra vadinimi žetonais.
* Matematiškai, baigtinis automatas, bazuojamas reguliaria gramatika.
* Leksinis analizatorius priima ASCII simbolius, atmeta iš jų tarpus, komentarus, padaro pradinę validaciją, jei yra klaidų – meta klaidą, jeigu viskas gerai – perskaitytas reikšmes paverčia token‘ais ir perduoda sintaksės analizatoriui.

**Išvardinkite leksinio analizatoriaus sudarymo būdus.**

* Parašyti formalų žetonų aprašą ir naudoti PĮ, kuri sudaro lentelinį leksikos analizatorių šiam aprašui. Lex Unix
* Suprojektuoti būsenų diagramą, kuri aprašo žetonus ir parašyti programą, kuri įgyvendina šią būsenų diagramą.
* Suprojektuoti būsenų diagramą, kuri aprašo žetonus ir parašyti lentelinį šios diagramos įgyvendinimą.

**Kas yra būsenų diagrama leksiniame analizatoriuje**

* Tai diagrama turinti išeities kalbos kiekvieno simbolio kiekvienos būsenos perėjimą

**Kokie yra du skirtingi sintaksinės analizės tikslai?**

* Surati visas sintaksines klaidas. Kiekvienai klaidai išduoti atitinkamą diagnostinį pranešimą.
* Sukurti gramatinį medį,

**Paaiškinkite dvi gramatikos charakteristikas, kurios neleidžia gramatikai būti iš viršaus žemyn gramatinio analizatoriaus baze.**

* Jei gramatika turi tiesioginę ar netiesioginę kairiąją rekursiją,
* Porinio nesusijungimo nebuvimas

**Paaiškinkite, kodėl kompiliatoriuose naudojami analizės algoritmai, kurie dirba tik su gramatikų poaibiu.**

* Jų laikas yra tiesinis.

**Pristatykite tris LR gramatinių analizatorių privalumus**

* Jie dirba beveik su visų programavimo kalbų gramatikomis.
* Jie dirba platesnėje gramatikų klasėje, nei kiti iš apačios į viršų algoritmai, bet lygiai taip pat efektyvūs, kaip ir kiti algoritmai
* Aptinka klaidas taip greitai, kaip tik įmanoma

**Koks yra ryšys tarp LR ir LL gramatinių analizatorių?**

* LR gramatikų klasė yra gramatikų, nagrinėjamų LL parseriai, super aibė

# Vardai

**Ką vadiname rezervuotu žodžiu?**

* *Rezervuotas žodis* - specialus žodis, kuris negali būti naudojamas kaip vartotojo apibrėžtas vardas
* Problema su rezervuotais žodžiais: daug susidūrimų (COBOL turi 300 rezervuotų žodžių!)
* Iš anksto apibrėžti (predefined) vardai: bibliotekų paprogramės. Jie turi prasmę, bet vartotojas jiems gali suteikti savo prasmę

**Ką vadiname raktiniu žodžiu?**

*Raktinis* žodis – tai žodis, kuris yra specialus tik tam tikrame kontekste (Fortran);

* **Real Integer** (**Real** *duomenų tipas, todėl* **Real** *- raktinis žodis)* **Integer = 3.4** (***Integer*** *– kintamasis )*

**Kas yra kintamojo l-reikšmė? Kas yra kintamojo r-reikšmė?**

- **l-reikšmė** – tai kintamojo adresas

- **r-reikšmė** – tai kintamojo reikšmė

**Kokius atributus turi kintamuosius?**

1. Tipas - svarbiausias
2. Vardas – vienas iš svarbesniųjų
3. Adresas
4. Reikšmė
5. Gyvavimo laikas
6. Galiojimo sritis

**Kokie yra statinių kintamųjų privalumai ir trūkumai?**

* Statinis – susiejamas su atminties ląstele iki vykdymo ir lieka susietas su pačia ląstele per visą vykdymą (C ir C++ static kintamieji)
  + Privalumai: efektyvumas (tiesioginis adresavimas), paprogramių istorijos palaikymas
  + Trūkumas: lankstumo trūkumas (nėra rekursijos), atmintimi negalima pasidalinti
* **static** kintamieji C++ klasėje nėra tas pats, kas **static** kintamieji už klasės ribų. Maišatis

**Kokios yra skaliarinių kintamųjų kategorijos pagal gyvavimo laiką?**

* Statinis
* Dėklo dinaminis
* Išreikštas kupetos dinaminis
* Numanomas kupetos dinaminis

**Apibrėžkite gyvavimo laiką ir galiojimo sritį. Koks yra skirtumas tarp jų?**

* Kintamojo galiojimo sritis – tai sakiniai, kuriuose jis matomas
* Kintamasis yra matomas sakinyje, jei į kintamąjį galima kreiptis
* Kalbos galiojimo srities taisyklės nustato, kaip vardų nuorodos susiejamos su kintamaisiais
* Programos vieneto nelokalūs kintamieji yra tie, kurie jame matomi, bet nedeklaruoti
* Kintamojo gyvavimo laikas – tai laiko intervalas, kada jam buvo išskirtas atminties blokas.
* Skirtumas – skirtingos koncepcijos. Galiojimo sritis – erdvinė koncepcija. Gyvavimo laikas – laikinė koncepcija.

**Kokie yra statinės galiojimo srities privalumai ir trūkumai?**

Privalumai:

* Veikia gerai daugelyje situacijų

Trūkumai:

* Daugelyje atvejų per daug pasiekiamas
* Kai programa vystoma, pradinė struktūra sugriaunam ir lokalūs kintamieji tampa globaliais; paprogramės irgi traukia link globalių

**Kokie yra dinaminės galiojimo srities privalumai ir trūkumai?**

* Privalumas – patogumas
* Trūkumai:
  + Kai paprogramė vykdoma, jos kintamieji yra matomi visiems paprogramės kreipiniams
  + Neįmanomas statinis tipų tikrinimas
  + Blogas skaitomumas – statiškai neįmanoma nustatyti kintamojo tipo

**Kas yra sakinio gyvavimo aplinka?**

Sakinio gyvavimo aplinka yra vardų, kurie matomi tame sakinyje, rinkinys

Statinio akiračio kalbose lokalūs kintamieji ir visi kiti apimančio akiračio kintamieji

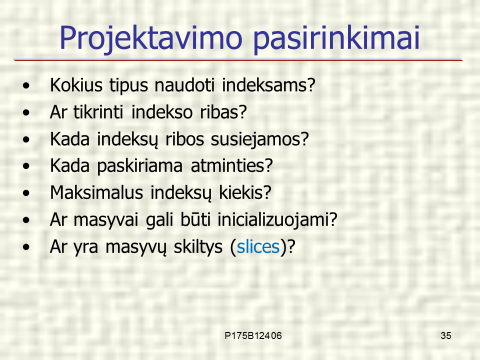
Paprogramė yra aktyvi, jei ji pradėjo vykdymą, bet dar nepabaigė

Dinaminio akiračio kalbose visi lokalūs kintamieji ir visi matomi visų aktyvių paprogramių kintamieji

# Tipai

**Kokie yra projektavimo pasirinkimai dėl masyvų?**

* Masyvas – tai homogeninių elementų junginys, kuriame kiekvienas elementas įvardijamas jo pozicija, pradedant nuo pirmojo elemento.
* Atskirą elementą pasiekiame, naudodami indeksą (subscript)
* JavaScript, Python, Ruby kintamieji – betipės nuorodos į objektus. Masyvo elementai gali rodyti į skirtingo tipo objektus



**Kas yra heterogeninis masyvas? Kokiose kalbose jis yra įmanomas?**

Heterogeninis masyvas – tai masyvas, kuriame elementai gali būti skirtingų tipų, nes kintamieji yra betipės nuorodos į objektus.  
Tai įmanoma: Perl, Python, JavaScript, and Ruby

**Ką vadiname dantytu masyvu?**

Dantytas masyvas turi eilutes su kintančiu elementų skaičiumi.

**Kokie projektavimo pasirinkimai dėl eilučių?**

Ar tai primityvus tipas ar specialus masyvo atvejis?   
Ar eilutės ilgis turėtų būti statinis ar dinaminis?

**Kas yra prieigos funkcija masyvui?**

* Prieigos funkcija atvaizduoja indekso išraiškas į adresą masyve
* Prieigos funkcija vienmačiam:

adresas(list[k]) = adresas (list[žemiausioji riba]) + ((k – žemiausioji riba) \* elemento dydis)

**Kas yra asociatyvus masyvas ir kokie jo privalumai?**

Asociatyvus masyvas yra nesutvarkytas duomenų rinkinys, kuriame saugomas raktas ir reikšmė. Raktai yra unikalūs.

Privalumai tokie, kad yra patogu paieškai, kadangi dažniausiai būna implementuota naudojant maišos lenteles. Labai geras panaudojimo būdas, kai raktas yra suporuotas su reikšme. Pvz žodyne žodis ir jo paaiškinimas.

**Kokios asociatyvaus masyvo ypatybės PHP kalboje?**

PHP masyvas gali būti ir paprastas, ir asociatyvusis. Raktu gali būti sveikieji skaičiai ir eilutės.

**Į kokias kategorijas skirstomi masyvai pagal atmintį ir indeksų saistymą?**

* *Statiniai*:masyvo ribos yra statiškai susiejamos ir atminties paskiriama prieš vykdymą
  + Privalumas: efektyvumas (nėra dinaminio priskyrimo) c, c++, kurie turi static modifikatorių
* *Fiksuotas dėklo dinaminis*: masyvo ribos yra statiškai susiejamos ir bet atminties paskiriama vykdymo metu
  + Privalumas: erdvės efektyvumas c, c++ be static
* *Dėklo dinaminis*: indeksų ribos susiejamos dinamiškai ir atminties išskiriama dinamiškai vykdymo metu
  + Privalumas: lankstumas – masyvo dydžio nereikia žinoti iki masyvo naudojimo Ada
* *Fiksuotas kupetos dinaminis*: panašiai, kaip ir dėklo dinaminis, tik atmintis iš kupetos C# ArrayList, Java
* *Kupetos dinaminis*: susiejimas indeksų ir atminties dinaminis bei gali keistis daug kartų
  + Privalumas: lankstumas – masyvai gali augti ar trauktis vykdymo metu Perl JavaScript, Python, Ruby

**Kuo pasižymi masyvo inicializavimas Ada kalboje?**

Ada leidžia įrašus inicializuoti apjungtais literalais

**Kokie yra daugiamačio masyvo saugojimo būdai atminyje? Pateikite kalbų atstovus**

????????

# Išraiškos

**Ką apibrėžia operatorių asociatyvumo taisyklės?**

Jie nustato kuria tvarka yra vykdomi šalia esantys operatoriai.

(Pagal skaidres): Nustato eilę, kuria „gretimi“ vienodos viršenybės operatoriai yra įvertinami.

**Kas yra operatorių užklojimas?**

Tai tokių operatorių kaip(+,-,\*,/)reikšmės suteikimas/pakeitimas.C++ ir C# leidžia užkloti vartotojo sugalvotus operatorius.

Trumpiau: kai operatorius naudojamas daugiau nei vienam tikslui.

**Kas yra ne asociatyvus operatorius?**

Tai kai operatoriai negali būti kartu išraiškoje (dažniausiai dėl to, kad jie sukuria kitokios struktūros objektą, su kuriuo jau kiti operatoriai paprastai dirbti negali)

**Kas yra šalutinis funkcijos efektas?**

Kai funkcija keičia dviejų krypčių parametrą arba ne lokalų kintamąjį

**Kokia yra problema, susieta su funkcijos šalutiniu efektu?**

Kai funkcija išraiškoje keičia kitą išraiškos operandą.

**Kada atliekamas priverstinis tipo keitimas? Koks terminas?**

Norint paversti objektą iš vieno tipo į kitą. Pavyzdžiui norint atspausdinti int reikšmę, reikia ją pirmiausia paversti į string. Sąvokos: Coercion, Casting, type conversion

**Kas yra nuorodų skaidrumas?**

Tai kai pakeitus bet kurias dvi programos išraiškas (kurios įgyja tą pačią reikšmę) viena su kita, programos rezultatai nepakinta.

**Kas yra sutrumpintas įvertinimas?**

Išraiška, kai rezultatas gaunamas neįvertinant visų operandų. Pvz (a \* 0) \* (17 \* b) 17\*b nebus įvertinta, nes rezultatas ir taip 0. Taip pat naudajamas su loginiais operatoriais c, c++, Java, C#(&&, ||)

**Koks yra galimas trūkumas, kai į priskyrimo operatorių žiūrime kaip į aritmetinį operatorių?**

Neįprasta vykdymo eilė (tvarka)

**Kokiu terminu įvardijamas išreikštas tipo pakeitimas?**

Casting

(Skaidrėse) Coercion – numanoma tipų konversija

# Sakinių valdymas

**Koks yra vienintelis projektavimo pasirinkimas dėl valdymo struktūrų?**

Ar valdymo struktūra gali turėti kelis įėjimus?

**Kokias išrinkimo sakinių kategorijas žinote?**

Dviejų kelių pasirinkimas (if => else =>)

Daugelio kelių pasirinkimas (iš daug variantų galima pasirinkti vieną)

**Kuomet valdymo išraiška talpinama į skliaustus?**

Jei nenaudojamas rezervuotas žodis then ar kitas sintaksinis žymeklis, valdymo išraiška rašoma skliaustuose

**Kokiose kalbose valdymo išraiška privalo būti loginė?**

Ada, Java, Ruby, C#

**Paaiškinkite kuo C# switch sakinys saugesnis, nei Java**

C# turi goto sakinį.

Kiekvienas pasirenkamas segmentas C# kalboje turi baigtis besąlygine šaka goto arba break, todėl neleidžia vykdyti daugiau nei vieno segmento.

**Koks skirtumas tarp for sakinių C++ ir Java?**

Java valdymo išraiška privalo būti loginė (boolean)

**Ką atlieka range funkcija Python?**

Grąžina eilę reikšmių, nurodytų range funkcijoje. Range(5) grąžina 0,1,2,3,4,5

**Kokie yra skirtumai tarp break sakinių C++ ir Java?**

C++ break išėjimas yra nežymėtas, java jis yra žymėtas

**Kokia yra bazinė Dijkstra apsaugotų komandų idėja?**

Aprašyti programą apsaugotų komandų kalba prieš padarant tai praktiška programavimo kalba, kad būtų galima patikrinti programos korektiškumą.