PROGRAMAVIMO KALBŲ PARADIGMOS

Vitalijus Lovkis, Deividas Bružas,
darbo vadovai lekt. Kučinskas Gintaras, doc. dr. Kučinskienė Jurga
Klaipėdos valstybinė kolegija

ANOTACIJA

Šiame straipsnyje analizuojama programavimo kalbų paradigmų svarba ir taikymas. Straipsnyje supažindinama su bendra sąvoka, kas yra programavimo paradigma, apžvelgiami pagrindiniai programavimo paradigmų tipai, jų privalumai ir skirtumai. Straipsnyje analizuojama informacija, kuriose srityse yra pritaikomos konkrečios programavimo paradigmos atsižvelgiant į projektų tikslus.
Raktiniai žodžiai: programavimo paradigma, funkcinis programavimas, objektinis programavimas.

ĮVADAS

Pasak Marijos Upson (Upson, 2022), šiuo metu pasaulyje egzistuoja apie devyni tūkstančiai skirtingų programavimo kalbų. Dauguma žmonių susiduria su sunkumais, kaip pasirinkti tinkamą programavimo kalbą pagal projekto sritį. Bet kuris programuotojas ar programinės įrangos kūrėjas turėtų žinoti apie įvairias programavimo paradigmas, nes taip bus lengviau pasirinkti geriausią strategiją programavimo iššūkiui. Programuotojai, naudodamiesi programavimo paradigmomis, gali supaprastinti programinės įrangos kūrimo procesą ir tokiu būdu pateikti efektyvius bei kokybiškus sprendimus. Programuotojai dažnai pasirenka vieną ar daugiau iš programavimo paradigmų, kurias naudos savo programinės įrangos kūrimo projektuose, nes kiekviena iš šių paradigmų turi unikalių savybių ir pranašumų.

Skirtingose taikymo srityse naudojamos įvairios programavimo paradigmos, nes programinės įrangos kūrimo pagrindas formuojamas atsižvelgiant į norimo rezultato reikalavimus. Pavyzdžiui, finansų srityje dažniausiai naudojamas procedūrinis arba objektinis programavimas, nes ten dažnai reikia taikyti skaičiavimo algoritmus. Medicinoje dažniausiai taip pat naudojamas objektinis programavimas, kadangi reikia generuoti pacientų duomenis, kurie vėliau skelbiami konkrečiose duomenų bazėse. Atliekant internetinių svetainių programavimą dažniausiai yra naudojamas funkcinis ir objektinis programavimas kartu.

Tyrimo objektas – programavimo kalbų paradigmos.

Tikslas – išanalizuoti ir įvertinti programavimo kalbos pasirinkimą, priklausomai nuo projekto srities.

Tyrimo metodai: atlikta atpasakojamoji bei aprašomoji apžvalga, siekiant pateikti teminę analizę, apžvelgti koncepcinius karkasus.

1. PROGRAMAVIMO PARADIGMOS

Programavimo paradigma – tai būdas išspręsti problemą naudojant tam tikrą programavimo kalbą arba taip pat galima sakyti, kad tai būdas išspręsti problemą naudojant įrankius ir metodus, kurie mums yra prieinami laikantis tam tikro požiūrio (Rani, 2022).

1.1. Procedūrinis programavimas

Procedūrinis programavimas – tai linijinė idioma, kai kūrėjas rašo nuoseklią kodo teiginių seką, kuri yra vykdoma eilute, viena po kitos, taikant metodą iš viršaus į apačią, panašiai kaip laikantis paprasto gaminimo recepto (Hodges, 2019).

Procedūrinis programavimas suskaido programą į mažesnius kodo vienetus, kuriuos galima lengviau naudoti pakartotinai, prižiūrėti ir išbandyti. Šis metodas pabrėžia laipsnišką problemų sprendimo metodą, kai problema suskaidoma į mažesnes, lengviau valdomas užduotis, kurias galima atlikti nuosekliai. Procedūrinės programavimo kalbos, tokios kaip „C“, „Fortran“, „Pascal“ ir „Basic“, buvo vienos iš pirmųjų programavimo kalbų ir vis dar naudojamos šiandien. Ši programavimo paradigma plačiai naudojama daugelyje sričių, pavyzdžiui, ***mokslo ir inžinerijos programose, operacinėse sistemose ir žaidimų varikliuose***.

Pagrindinės procedūrinio programavimo ypatybės yra (Bhatia, 2022):

- **Iš anksto nustatytos funkcijos** – funkcija, kuri yra apibrėžta iš anksto, yra instrukcija, identifikuojama pavadinimu. Funkcijos, kurios yra apibrėžiamos iš anksto, integruojamos į aukštesnį programavimo kalbų lygį, bet dažniausiai tos funkcijos yra gaunamos iš įvairių bibliotekų, bet ne iš programų.
- **Sistemų grupavimas** – yra tada, kai dvi skirtingos sistemos turi dvi skirtingas užduotis, bet yra sugrupuotos, kad pirmiausia būtų užbaigta didesnė užduotis. Tuomet kiekviena sistemų grupė savo užduotis užbaigtų vieną po kitos, kol visos užduotys būtų baigtos.
- **Parametrų perdavimas** – tai mechanizmas, naudojamas parametrams perduoti funkcijoms, paprogramėms ar procedūroms. Parametrų perdavimą galima atlikti naudojant „perdavimą pagal vertę“, „perdavimą pagal nuorodą“, „perdavimą pagal rezultatą“, „perdavimą pagal vertę-rezultatą“ ir „perdavimą pagal pavadinimą“.

1.2. Objektinis programavimas

Objektinis programavimas, sutrumpintai OOP, yra kompiuterių programavimo paradigma informatikos srityje. Skirtingai nuo tradicinio požiūrio, kuriame programa gali būti vertinama kaip funkcijų rinkinys arba tiesiog kaip kompiuteriui skirtų instrukcijų sąrašas, objektinis programavimas siūlo, kad kompiuterinė programa gali būti laikoma sudaryta iš atskirų vienetų rinkinio, arba objekto, kurie veikia vienas kitą. Kiekvienas objektas turi galimybę apdoroti duomenis, gauti pranešimus ir

siųsti pranešimus kitiems objektams. Į kiekvieną daiktą galima žiūrėti kaip į atskirą, mažą veikėją ar mašiną, kuri atlieka tam tikrą funkciją (Pandey, 2015).

Taigi, kad programavimo kalba veiktų kaip objektinė programavimo kalba, ji turi turėti šias į objektą orientuotas funkcijas:

- **Polimorfizmas** – kai kas nors gali būti kelių skirtingų formų. Informatikoje ji apibūdina sąvoką, kad skirtingų tipų objektus galima pasiekti per tą pačią sąsają. Kiekvienas tipas gali pateikti savo nepriklausomą šios sąsajos realizaciją (Janssen, 2021).
- **Paveldėjimas** – tai mechanizmas, kuriuo galima išvesti klasę iš kitos klasės ir taip sukurti klasių, turinčių bendrą atributų ir metodų rinkinį, hierarchiją. Jį galite naudoti norėdami deklaruoti įvairių rūšių išimtis, pridėti pasirinktinę logiką prie esamų struktūrų ir net atvaizduoti savo srities modelį į duomenų bazę (Janssen, 2017).
- **Inkapsuliavimas** – pagal apibrėžtį inkapsuliavimas apibūdina duomenų ir metodų, kurie dirba su tais duomenimis, sujungimą į vieną vienetą, pavyzdžiui, „Java“ klasę. Šią sąvoką dažnai naudojame norėdami paslėpti objekto vidinę reprezentaciją arba būseną nuo išorės.

1.3. Funkcinis programavimas

Funkcinis programavimas yra kompiuterinių programų kūrimas, naudojant išraiškas ir funkcijas, nematuojant būsenos ir duomenų. Laikantis anksčiau išvardintų apribojimų, funkcinis programavimas siekia parašyti kodą, kuris būtų aiškesnis, suprantamas ir atsparesnis klaidoms. Tai pasiekiama vengiant naudoti srauto valdymo teiginius (for, while, break, continue, go to), kurie apsunkina kodo sekimą. (Aryan, 2020).

Funkcinis programavimas, kaip viena iš pagrindinių paradigmų, pabrėžia funkcijų naudojimą problemoms spręsti. Funkcinis programavimas yra svarbus, nes juo skatinamas moduliavimas, pakartotinis naudojimas ir lengviau galima pagrįsti programas (Hughes, 1989). Tai deklaratyvus programavimo stilius, o tai reiškia, kad apibrėžiama, kas norima pasiekti, o ne kaip tai padaryti. Šitoje programavimo paradigmoje funkcijos yra pirmos klasės, tai reiškia, kad funkcijas galima perduoti kaip argumentus kitoms, taip pat grąžinti jas iš kitų funkcijų ir deklaruoti kaip kintamuosius. Funkcinis programavimas taip pat paprastai vengia kintančios būsenos ir šalutinių poveikių, o pirmenybę teikia nekintamoms duomenų struktūroms ir grynoms funkcijoms. Dėl to gali būti sukurtas kodas, kurį lengviau suprasti, išbandyti ir prižiūrėti. Kai kurios populiarios funkcinio programavimo kalbos yra „Haskell“, „Clojure“, „Scala“ ir „F#“. Tačiau daugelis kalbų, įskaitant „JavaScript“, „Python“ ir „Ruby“, taip pat palaiko funkcinio programavimo funkcijas.

Funkciniame programavime duomenys laikomi nekeičiamais, tai yra kai kintamajam priskiriama reikšmė, kurios negalima keisti. Naujoms reikšmėms kurti naudojamos funkcijos. Funkcinis programavimas pasižymi ypatybėmis, kurios leidžia detaliau suprasti, kodėl ši

programavimo paradigma remiasi idėja, kad sukurtos funkcijos yra nepriklausomos viena nuo kitos ir gali būti paleidžiamos bet kuria eiga (Posa, 2022).

Naujoms reikšmėms kurti naudojamos funkcijos, kurios priima įvesties reikšmes ir generuoja naujas išvesties reikšmes. Šis modelis vadinamas „**nekeičiamu“**.

Funkciniame programavime aukštesnės eilės funkcijos laikomos pirmos klasės objektais, kurie suteikia vartotojams galimybę juos naudoti kaip argumentus kitose funkcijose, grąžinti kaip rezultatus iš kitų funkcijų ir saugoti kintamuosiuose.

Rekursijos metodai yra plačiai naudojami funkciniame programavime, kurie atlieka duomenų struktūrų kartojimą ir skaičiavimą.

Grynųjų funkcijų metodas naudojamas funkciniame programavime, kuris turint tos paties įvesties vertę leidžia generuoti visada tą pačią išvestį ir neturint šalutinių poveikių. Dėl to grynųjų funkcijų testavimas ir argumentavimas yra paprastesnis funkciniame programavime.

Grynojo funkcinio programavimo metodas skatina ir leidžia sudėtingesnių funkcijų kūrimą iš mažesnių, pakartotinai naudojamų funkcijų.

Tingaus vertinimo metodas leidžia funkcinio programavimo kalbose išraiškas vertinti tik tada, kai jų reikia, o ne iš anksto (angl. *eagerly*).

Tipo išvados metodas naudojamas daugelio funkcinio programavimo kalbose, kad būtų išvengta aiškių tipo anotacijų.

1.4. Loginis programavimas

Loginis programavimas yra programavimo paradigma, kuri remiasi formalios logikos principais. Loginėje programavimo kalboje aprašomas yra ryšys tarp įvairių faktų ir taisyklių, o kalba naudoja logines išvadas, kad gautų naujus faktus ar taisykles iš jūsų pateiktų. Dėl to jis ypač tinkamas tokioms programoms kaip ekspertų sistemos, kuriose norite užkoduoti taisyklių rinkinį ir leisti, kad sistema naudotų jas išvadoms ar rekomendacijoms daryti. Viena iš pagrindinių loginio programavimo ypatybių yra ta, kad jis leidžia apibrėžti ryšius tarp objektų tokiu būdu, kuris nepriklauso nuo jokio konkretaus įgyvendinimo. Dėl to loginis programavimas yra ypač naudingas problemoms, kurias sunku išspręsti naudojant tradicinius procedūrinius arba objektinio programavimo metodus. Kai kurios populiarios loginio programavimo kalbos yra „Prolog“, „Mercury“ ir „Oz“. Šios kalbos naudojamos įvairiose programose, įskaitant natūralios kalbos apdorojimą, ekspertų sistemas ir duomenų bazių programavimą (Bramer, 2013).

Taip pat loginiame programavime yra ir kitos septynios programavimo ypatybės (Fernandez, 2014):

- **Deklaratyviame programavime** yra nurodomas programos tikslas, o ne jo įgyvendinimas. Tam naudojami loginiai teiginiai ir taisyklės, nurodančios ryšius tarp įvairių programos dalių.

- **Loginis išvedimas** loginiame programavime apibrėžia taisykles ir teiginius loginės programos išvadoms daryti. Tokiu atveju programa gali pati daryti išvadas dėl naujos informacijos, remdamasi anksčiau žinomais faktais ir dėsniais.
- **Suvienodinimo metodas** loginiame programavime yra esminė procedūra, apimanti kintamųjų ir reikšmių atitikimą ir sujungimą. Tai leidžia programai susinchronizuoti kelis duomenų bitus ir iš jų daryti tam tikras išvadas.
- **Grįžimo atgal** (angl. *Backtracking*) metodas yra naudojamas įvairioms alternatyvoms ir keliams tirti. Jeigu programa susiduria su sunkumais, ji gali apsisukti ir išbandyti kitą suprogramuotą kelią.
- **Rekursijos metodas** plačiai ir dažnai naudojamas loginiame programavime, kai funkcija arba taisyklė gali iškviesti save pačią, kad problema būtų išspręsta šiuo būdu. Tai leidžia sudėtingai ir glaustai atsakyti į algoritminius veiksmus.
- **Taisyklėmis pagrįstas** programavimo metodas loginiame programavime dažnai dar įvardijamas kaip taisyklėmis grindžiamu programavimo metodu, kadangi programa sudaryta iš loginių taisyklių ir teiginių, nurodančių, kaip įvairūs elementai susiję tarpusavyje.
- **Nedeterminuotas metodas** loginiame programavime leidžia nedeterminuoti elgseną, kai programos išvestis gali skirtis priklausomai nuo įvesties ir joje esančių taisyklių.

2. PROGRAMAVIMO KALBŲ PASIRINKIMO VERTINIMO KRITERIJAI

Apžvelgiant skirtingas programavimo kalbas, būtina atsižvelgti į kriterijus, pagal kuriuos jas būtų galima vertinti. Pagrindiniai kriterijai programavimo kalbų apžvalgai yra ***požiūris į problemų sprendimą, duomenų tvarkymo būdą, programų sudėtingumą***.

Naujiems programuotojams problemų sprendimas dažnai yra sunkiausiai įgyjamas įgūdis. Neretai pradedantieji programuotojai lengvai išmoksta sintaksę ir programavimo sąvokas, tačiau, kai bando ką nors savarankiškai programuoti, jie tuščiai žiūri į teksto redaktorių ir nežino, nuo ko pradėti (Trautman, be datos). Požiūris į *problemų sprendimo kriterijų* leidžia suprasti, kuri kalba naudoja tam tikrą sprendimo metodiką.

Duomenų apdorojimas – tai neapdorotų duomenų rinkimo ir pavertimo, tinkamai naudoti informaciją, metodas. Jį paprastai žingsnis po žingsnio atlieka organizacijos: duomenų mokslininkų ir duomenų inžinierių komanda. Neapdoroti duomenys renkami, filtruojami, rūšiuojami, apdorojami, analizuojami, saugomi, o tada pateikiami suprantamu formatu (Duggal, 2023). *Duomenų tvarkymo būdo* kriterijus leidžia sužinoti, kuri programavimo kalba gali efektyviau apsaugoti duomenis.

Programų sudėtingumo kriterijus leidžia sužinoti, kuri programavimo kalba labiau tinka kurti projektus pagal jų apimtis. Programos, kuri turi būti vykdoma skaičiavimo mašinoje, sudėtingumas

priklauso nuo mašinos skaičiavimo galios ir jos išteklių. Tradiciniai programų sudėtingumo matai yra laikas, erdvė ir programos apimtis (Iyengar, Parameswaran, & Fuller, 1982).

Programavimo kalbų skirtumai pagal pasirinkimo vertinimo kriterijus. Dvi pagrindinės programavimo paradigmos: procedūrinio programavimo ir į objektus orientuoto programavimo – turi skirtingus ***požiūrius į problemų sprendimą***. Procedūrinio programavimo metodika, taikanti „iš viršaus į apačią" strategiją, programą suskirsto į valdomus vienetus vadinamus funkcijomis. Akcentuojami veiksmai, kuriuos reikia atlikti norint atlikti tam tikrą užduotį. Objektinis programavimas, priešingai, vadovaujasi „iš apačios į viršų" metodu, suskirstydamas programą į atskirus vienetus vadinamus objektais. Pabrėžiami duomenys ir su tais duomenimis susijusi elgsena.

Dar vienas svarbus skirtumas – procedūrinio programavimo ir objektinio programavimo ***duomenų tvarkymo būdas***. Kadangi procedūrinio programavimo metu duomenų negalima tinkamai paslėpti, jis yra mažiau saugus. Tačiau objektiniame programavime duomenys paslepiami naudojant tokius prieigos žymenis kaip privatus (angl. private), viešas (angl. public) ir apsaugotas (angl. protected), todėl jis yra saugesnis. Be to, paveldėjimas nėra procedūrinio programavimo sąvoka, o paveldėjimas yra pagrindinis objektinio programavimo komponentas. Todėl objektai gali perimti kitų daiktų bruožus ir savybes, o tai gali lemti nemažą pakartotinį kodo panaudojimą.

Programų, kurias galima sukurti naudojant kiekvieną programavimo paradigmą, ***sudėtingumas*** taip pat skiriasi. Kuriant vidutinio dydžio programas, kurių pagrindinis dėmesys yra skirtas tam tikroms operacijoms atlikti, gali būti naudingas procedūrinis programavimas. Priešingai, objektinis programavimas yra tinkamas kuriant didžiules, sudėtingas programas pagrįstas duomenimis ir su jais susijusia elgsena. Be to, procedūrinis programavimas apsunkina naujų duomenų ir funkcijų pridėjimą nei objektinis programavimas. Apskritai, problemos, kurias reikia išspręsti, tipas ir kuriamos programos sudėtingumas lemia, ar reikia naudoti procedūrinį ar objektinį programavimą.

Programavimo paradigmos, tokios kaip loginis programavimas ir funkcinis programavimas, pasižymi unikaliomis savybėmis, kurios skiriasi viena nuo kitos.

Požiūris į problemos sprendimą loginiame programavime akcentuoja programos būseną ir elgesio aprašymą, tai reiškia, kad loginio programavimo dėmesys yra perėjimas iš vienos būsenos į kitą. O funkcinio programavimo akcentavimas yra skirtas programos veiksmų sekai, kai reikia atlikti kitus veiksmus prieš atliekant svarbiausią užduotį ir tuo metu mažiau dėmesio skiriama programos būsenai.

Duomenų tvarkymo būdas ganėtinai skirtingas tarp loginio programavimo ir funkcinio. Loginis programavimas dažniausiai naudoja globalius kintamuosius, kurie sąveikauja kaip parametrai funkcijai, kad būtų išsaugoma būsena.

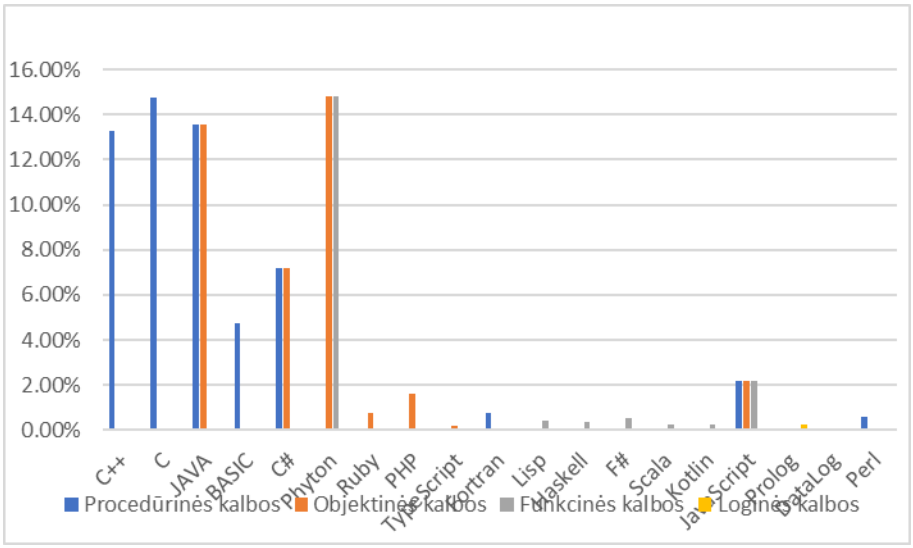
Atsižvelgiant į programos sudėtingumą, loginio programavimo projektai yra vidutinės apimties, kurie susideda iš tam tikrų perėjimų ir būsenų. Funkcinio programavimo projektai

dažniausiai yra nedidelės apimties, atsižvelgiant į tai, kad programos veikimo principas susideda iš įvykių sekos. Verta paminėti, kad funkcinis programavimas gali būti adaptuotas ir didelės apimties projektuose, kadangi funkcijos gali būti konvertuojamos į sudėtingesnes funkcijas.

„Neigimas kaip nesėkmė yra galinga idėja, svarbi loginio programavimo semantikai“ (Clark, 1979).

3. PROGRAMAVIMO KALBŲ PRITAIKYMAS

1 pav. nurodo programavimo kalbų naudojimo dažnumą, atsižvelgiant į skirtingas paradigmas. Programavimo kalbos pritaikomos atsižvelgiant į kuriamo projekto idėjas ir sritis.



1 pav. Programavimo kalbų pasirinkimo dažnumas
Šaltinis: sudaryta autorių pagal Tiobe korporacija, 2023

Pavyzdžiui, **procedūrinės kalbos** paradigmoje dažniausiai yra naudojamos C šeimos kalbos ir „JAVA“ programavimo kalba. Operacinės sistemos, sisteminis programavimas, moksliniai skaičiavimai ir automatizavimas – tai tik kelios programinės įrangos kūrimo sritys, kuriose plačiai taikomas procedūrinis programavimas.

Sisteminis programavimas yra viena iš sričių, kurioje procedūrinis programavimas yra plačiai naudojamas. Programavimo metodai, vadinami procedūriniu programavimu, plačiai naudojami kuriant tokias operacines sistemas kaip „Linux“, „Windows“ ir „MacOS“. Procedūrinis programavimas siūlo būdą, kaip veiksmingai valdyti šioms sistemoms reikalingus aparatinės įrangos išteklius, kurie reikalauja griežtos tų išteklių kontrolės. Procedūrinio programavimo metodai naudojami operacinės sistemos branduoliui, kuris yra pagrindinė operacinės sistemos sudedamoji dalis. Branduolys atsakingas už užduočių planavimą, aparatinės įrangos išteklių paskirstymą ir saugumo bei apsaugos funkcijų užtikrinimą. (Brouwer, 2003)

Kita sritis, kurioje dažnai naudojamas procedūrinis programavimas yra automatizavimas. Kompiuterinių programų, skirtų pasikartojančioms užduotims, įskaitant duomenų apdorojimą ar ataskaitų rengimą, atlikimą, rašymą vadinama automatizavimu. Procedūrinis programavimas

išskaidydamas šiuos procesus į mažesnius, lengviau valdomus etapus, siūlo mechanizmą jiems automatizuoti. Procedūrinio programavimo metodai naudojami tokiomis kalbomis kaip „Python“ ir „Perl“ sukurtuose automatizavimo scenarijuose, skirtuose tokioms operacijoms, kaip duomenų analizei, žiniatinklio duomenų nuskaitymui ir failų apdorojimui atlikti (McKinney, 2022).

Kita programavimo paradigma, kuri dominuoja, yra **objektinio programavimo** paradigma (1 pav.). „OOP“ dažnai naudojama įvairiose programinės įrangos kūrimo srityse, pavyzdžiui, kuriant žaidimus.

Viena iš pramonės šakų, kurioje plačiai naudojama „OOP“ yra žaidimų kūrimas. Žaidimai – tai sudėtingos taikomosios programos, reikalaujančios didelio įsitraukimo ir sudėtingos elgsenos. „OOP“ suteikia galimybę klasifikuoti žaidimų objektus taip, kad jų elgesį būtų galima apibrėžti metodais ir savybėmis. Kad žaidimų kūrėjams būtų suteikta intuityvi ir lengvai pritaikoma aplinka, tokie žaidimų varikliai kaip „Unity“ ir „Unreal Engine“ pirmiausia remiasi „OOP“ idėjomis, kur ir yra naudojamos tokios populiarios programavimo kalbos kaip „C“, „C++“, „C#“.

Funkcinės kalbos – paradigmose populiariausios programavimo kalbos: „Python“, „Lip“, „Haskel“. Tinklalapių kūrimas yra viena iš sričių, kurioje plačiai taikomas funkcinis programavimas. Naudojant tokias funkcinio programavimo kalbas kaip „Haskel“, „Lip“ ir „Python“, kuriamos itin keičiamo dydžio ir saugios žiniatinklio programos. Šiose kalbose daug dėmesio skiriama nekeičiamumui, kuris reiškia, kad kartą sukurti duomenys negali būti pakeisti. Tai palengvina testavimą ir derinimą bei paspartina klaidų tvarkymą (Feldman, 2020).

Paskutinė paradigma – tai **loginis programavimas,** kuriame dažniausiai pabrėžiamas deklaratyvus programavimas, kai programos logika išreiškiama kaip loginių taisyklių ir apribojimų rinkinys. Dažniausiai loginio programavimo nauda pasižymi dirbtinio intelekto, duomenų bazių, programinės inžinerijos darbo srityse. Populiariausios programavimo kalbos, kurios naudoją loginio programavimo paradigmas yra „Prolog“ ir „ASP“.

IŠVADOS

1. Apžvelgus informacinius šaltinius nustatyta, kad išskiriamos keturios programavimo paradigmos: ***procedūrinis, objektinis, funkcinis ir loginis programavimas***. Skirtingos programavimo paradigmos leidžia kurti įvairias programų architektūras. Kiekviena iš šių paradigmų turi privalumų ir trūkumų, todėl pasirinkimas priklauso nuo konkretaus naudotojo poreikių ir viso projekto reikalavimų.

2. Išanalizavus programavimo kalbas nustatyta, kad programavimo kalbų pasirinkimo vertinimui naudojami pagrindiniai trys kriterijai: *požiūris į problemų sprendimą, duomenų tvarkymo būdas, programų sudėtingumas.* Įvertinus pagal šiuos kriterijus visas keturias paradigmas nustatyta, kad procedūrinis programavimas skirtas projektams, kurie yra vidutinės apimties ir turi vieną

užduotį, o objektinis programavimas skirtas didesnės apimties projektams, kurie turi tam tikrus objektus, savo elgseną ir keletą įvykių sekų. Loginis programavimas skirtas vidutinės apimties projektams, kuriems svarbi programos būsena ir elgsena, kas nėra būdinga funkciniam programavimui. Funkciniame programavime svarbiausias akcentas yra užduočių atlikimas ir įvykių sekos. Be to, funkcinio programavimo projektai dažniausiai yra didesnės apimties nei loginio programavimo projektai.

3. Programavimo kalbų pasirinkimą dažniausiai nulemia jų pagrindas, t. y. programavimo paradigma. Dažniausiai naudojamos tokios programavimo kalbos, kaip „Python“, „Java“, „C“ ir „C++“. Tačiau konkrečios kalbos pasirinkimą tam tikroje srityje nulemia tiek pati sritis, tiek programos konceptas. Pramonėje dažniausiai naudojamos „C++“ ir „Python“ programavimo kalbos, o žaidimams kurti dažniausiai tinkamiausios yra „C“ ir „C++“ programavimo kalbos. Dirbtinio intelekto srityje tikslingiausia naudoti „ASP“ ir „Prolog“ programavimo kalbas.

SUMMARY

Reach aim. analyse and evaluate the choice of programming language depending on the project area.

Research problem. Most people struggle with how to choose the right programming language for their project area. Any programmer or software developer should be aware of the different programming paradigms, as this will make it easier to choose the best strategy for the programming challenge. Programmers can simplify the software development process and thus provide efficient and high-quality solutions.

Key results and conclusion. This article analyses the importance and application of programming paradigms. It introduces the general concept of what a programming paradigm is, reviews the main types of programming paradigms, their differences, and advantages.

Keywords: Programming paradigm, functional programming, object-oriented programming.

LITERATŪRA

1. Aryan, A. (2020). *Introduction to Functional Programming: JavaScript Paradigms*. Prieiga per: https://www.toptal.com/javascript/functional-programming-javascript
2. Bhatia, S. (2022). *Procedural Programming* [Definition]. Prieiga per: https://hackr.io/blog/procedural-programming
3. Bramer, M. (2013). *Logic Programming with Prolog*, Second Edition. Springer.
4. Brouwer, A. (2003). *The Linux Kernel: An Introduction*.
5. Clocksin W, M. C. (2003). *Programming in Prolog*. Springer.

6. Duggal, N. (2023). *What Is Data Processing: Cycle, Types, Methods, Steps and Examples*. Prieiga per: https://www.simplilearn.com/what-is-data-processing-article
7. Feldman, R. (2020). *Elm in Action*. Manning.
8. Fernandez, M. (2014). General Features of Logic Programming Languages. *M. Fernández, Programming Languages and Operational Semantics* (p. 155-166). London: Springer.
9. Hodges, J. L. (2019). *Software Engineering from Scratch: A Comprehensive Introduction* Using Scala. Apress.
10. Iyengar, S., Parameswaran, N., & Fuller, J. (1982). *A measure of logical complexity of programs. Computer Languages*, p. 147-160.
11. Janssen, T. (2017). *OOP Concept for Beginners: What is Inheritance?* Prieiga per: https://stackify.com/oop-concept-inheritance/
12. Janssen, T. (2021). *OOP Concepts for Beginners: What is Polymorphism?* Prieiga per: https://stackify.com/oop-concept-polymorphism/
13. McKinney, W. (2022). *Python for Data Analysis*. O'Reilly Media.
14. Pandey, H. M. (2015). *Object-Oriented Programming C++ Simplified*. Laxmi Publications.
15. Posa, R. (2022). *Compare Functional Programming, Imperative Programming and Object Oriented Programming*. Prieiga per: https://www.digitalocean.com/community/tutorials/functional-imperative-object-oriented-programming-comparison
16. Rani, B. (2022). *Introduction of Programming Paradigms*. Prieiga per: https://www.geeksforgeeks.org/introduction-of-programming-paradigms/
17. Upson, M. (2022). *How many coding languages are there.* Prieiga per https://www.bestcolleges.com/bootcamps/guides/how-many-coding-languages-are-there/