

# **PROGRAMMĒŠANA**

**Augstākais mācību satura apguves līmenis**

**Centralizētā eksāmena programma**

**2023./2024. m. g.**

## SATURS

1. Eksāmena mērķis un adresāts .....	3
2. Vērtēšanas saturs.....	4
2.1. SR veidi un grupas .....	4
2.2. Satura moduļi .....	6
2.3. Izziņas darbības līmenis .....	6
3. Eksāmena darba uzbūve.....	8
4. Eksāmena piekļuves nosacījumi .....	9
5. Nepieciešamo resursu nodrošinājums.....	13
6. Vērtēšanas kārtība un kritēriji.....	14
6.1. Vērtēšanas kārtība .....	14
6.2. Vērtēšanas kritēriji .....	14
7. Palīgīdzekļi, kurus atļauts izmantot eksāmena laikā .....	15
Pielikumi .....	16
Programmēšanas labās prakses principi.....	17
Rīcības vārdu skaidrojums .....	18
Projekta izstrādes dokumentācijas snieguma līmeņu apraksts.....	25
Snieguma līmeņu apraksts programmatūras koda veidošanas posmam .....	30

## **1. EKSĀMENA MĒRĶIS UN ADRESĀTS**

Centralizētā eksāmena (turpmāk – eksāmens) mērķis ir novērtēt skolēnu sniegumu mācību priekšmetā atbilstoši Ministru kabineta 2019. gada 3. septembra noteikumiem Nr. 416 “Noteikumi par valsts vispārējās vidējās izglītības standartu un vispārējās vidējās izglītības programmu paraugiem” (turpmāk – standarts) un standarta 7. pielikumam “Plānotie skolēnam sasniedzamie rezultāti tehnoloģiju mācību jomā” optimālajā un augstākajā mācību satura apguves līmenī, identificēt un izvērtēt, cik lielā mērā ir apgūti plānotie sasniedzamie rezultāti (turpmāk – SR).

Eksāmena adresāts – skolēni, kuri ir apguvuši tehnoloģiju mācību jomas SR optimālajā un augstākajā mācību satura apguves līmenī atbilstoši mācību priekšmetu kursiem “Programmēšana I” un “Programmēšana II” (standarta 9. pielikums).

## 2. VĒRTĒŠANAS SATURS

Programmēšanas augstākā mācību satura apguves līmeņa eksāmena vērtēšanas saturu raksturo trīs kategorijas:

1. SR veids un grupa;
2. Satura modulis;
3. Izziņas darbības līmenis.

Tas nozīmē, ka katru eksāmena testelementu<sup>1</sup> raksturo noteikts SR veids un grupa, satura modulis un izziņas darbības līmenis.

### 2.1. SR veidi un grupas

Skolēnam plānoti triju veidu SR:

1. Zināšanas un izpratne;
2. Prasmju grupas;
3. Zināšanu, izpratnes, prasmju un ieradumu kombinācijas.

Katram SR veidam ir norādītas SR grupas (sk. 1. tabulu), kuras apkopo standartā noteikto mācību saturu un tiek pārbaudītas/mērītas eksāmenā.

Zināšanu un izpratnes pārbaudei eksāmenā iekļauti uzdevumi, kuros skolēni:

1. Atpazīst pamatalgoritmus un izprot to darbības principus;
2. Salīdzina programmatūras izstrādes modeļus un skaidro programmēšanas jēdzienus.

Prasmju apguvi raksturo trīs SR grupas:

1. Izstrādā programmatūru, tās prasību specifikāciju, izvēršanas plānu u. c.;
2. Lieto prasmes darbā ar informāciju, piemēram, projektējot datu bāzes struktūru, izstrādājot programmatūru, jēgpilni izmantojot esošo bibliotēku piedāvātās iespējas un programmēšanas valodas dokumentāciju, lai apgūtu programmēšanas valodas iespējas dotās problēmas risināšanā (t. sk. sintakses pārbaudē) u. c.;
3. Ievēro programmēšanas labās prakses principus (sk. 1. pielikumu) – vērtējumu par labās prakses principu lietošanu skolēns iegūst eksāmena uzdevumos, kuru vērtēšanas kritērijos norādīts par šo prasmju pārbaudi.

---

<sup>1</sup> Testelements ir uzdevums vai uzdevuma daļa, kas veidota, lai pēc kritērijiem vērtētu kādu konkrētu skolēnu darbības aspektu.

Zināšanu, izpratnes, prasmju un ieradumu kombināciju pārbaudei izmanto uzdevumus, kuri skolēniem liek atrisināt problēmu jaunā situācijā, turklāt to saturs var būt no jebkura satura moduļa (sk. 2. tabulu). Tie ir uzdevumi, kuros tiek kombinēti vairāki programmatūras dzīvescikla posmi: problēmas analīze, programmatūras specififikācija un darba plānošana, programmatūras produkta izstrāde.

1. tabula

SR veidi, grupas un to īpatsvars eksāmenā

SR veids un grupa		Īpatsvars, %	
Zināšanas un izpratne	Atpazīst pamatalgoritmus un izprot to darbības principus.	10 ± 5	
	Salīdzina programmatūras izstrādes modeļus un skaidro programmēšanas jēdzienus.		
Prasmju grupas	Izstrādā programmatūru, tās prasību specififikāciju, izvēršanas plānu u. c.	57 ± 5	68 ± 4
	Lieto prasmes darbā ar informāciju, piemēram, projektējot datu bāzes struktūru, izstrādājot programmatūru, jēgpilni izmantojot esošo bibliotēku piedāvātās iespējas un programmēšanas valodas dokumentāciju, lai apgūtu programmēšanas valodas iespējas dotās problēmas risināšanā (t. sk. sintakses pārbaudē) u. c.	8 ± 3	
	Ievēro programmēšanas labās prakses principus.	3 ± 1	
Zināšanu, izpratnes, prasmju un ieradumu kombinācijas	Kombinē vairākus programmatūras dzīvescikla posmus: problēmas analīzi, programmatūras specififikāciju un darba plānošanu, programmatūras produkta izstrādi, izvērtē drošības riskus	16 ± 4	

## 2.2. Satura moduļi

Satura moduļu īpatsvars eksāmenā (sk. 2. tabulu) ir proporcionāls kursa apguvei noteiktajam stundu skaitam.

2. tabula

Satura moduļi un to īpatsvars eksāmenā

Satura modulis	Īpatsvars, %
Datortīkls un datu bāze	20 ± 5
Programmatūras dzīvescikls (problēmas analīze, programmatūras specifikācija un darba plānošana)	10 ± 5
Objektorientētā programmēšana un ārējās bibliotēkas	33 ± 5
Datu struktūras, programmsaskarne (API)	33 ± 5

Eksāmena izstrādes procesā tiek saskaņots un nodrošināts sadaļu procentuālais sadalījums gan SR veidiem un grupām, gan satura moduļiem.

## 2.3. Izziņas darbības līmenis

Eksāmenā iekļautie uzdevumi grupēti četros izziņas darbības līmeņos, un to līmeņa noteikšanai izmanto SOLO jeb novēroto mācīšanās rezultātu taksonomiju. SOLO taksonomijā skolēna sniegums tiek raksturots, analizējot ideju jeb struktūrelementu skaitu un saišu kvalitāti starp šiem struktūrelementiem. Vispārīgs izziņas darbības līmeņu apraksts, kas piemērots eksāmenā, apkopots 3. tabulā.

**3. tabula**

**Izziņas darbības līmeņu raksturojums un to īpatsvars eksāmenā**

<b>Izziņas darbības līmenis un tā apraksts</b>		<b>Īpatsvars, %</b>
I	Atceras, nosauc atsevišķas idejas, izpilda īsas procedūras.	20 ± 5
II	Saista, skaidro, lieto zināšanas vai prasmes pazīstamās situācijās.	30 ± 5
III	Saista, skaidro, lieto zināšanas vai prasmes jaunās situācijās, demonstrējot izpratni (analizējot, pielietojot, pamatojot, salīdzinot).	30 ± 5
IV	Lieto zināšanas un prasmes situācijās ar augstu kompleksuma pakāpi.	11 ± 5

Katram līmenim atbilstošo uzdevumu īpatsvars noteikts, ievērojot eksāmena mērķi un galvenos vērtēšanas principus:

1. Pirmais princips – skolēnu grupai ar zemu un vidēju snieguma līmeni programmēšanā dota iespēja apliecināt savas zināšanas un prasmes pietiekami plašā satura jautājumu lokā, t. sk. uzdevumos, kas mēra zināšanu, izpratnes, prasmju un ieradumu kompleksu lietojumu;
2. Otrais princips – SR veidu “Zināšanas un izpratne” un “Prasmju grupas” vērtēšanai iekļauti testelementi, kas atbilst III izziņas darbības līmenim, tādējādi akcentējot izpratnes vērtēšanu;
3. Trešais princips – visi SR veida “Zināšanu, izpratnes, prasmju un ieradumu kombinācijas” vērtēšanai iekļautie uzdevumi ietver vismaz III līmenim atbilstošus testelementus.

### 3. EKSĀMENA DARBA UZBŪVE

Eksāmena darba uzbūvi, daļu īpatsvaru un izpildei paredzēto laiku var aplūkot 4. tabulā.

4. tabula

Eksāmena daļu īpatsvars un izpildei paredzētais laiks

Daļa	Uzdevumu skaits	Maksimālais punktu skaits	Daļas īpatsvars, %	Izpildes laiks, min
1. Datortīkls un datu bāze	2	20	17	70
2. Programmatūras dzīvescikls	5	20	17	
3. Objektorientētā programmēšana un ārējās bibliotēkas	2	38	31	80
4. Datu struktūras, programsaskarne (API)	2	42 (28 * 1,5)	35	80
<b>Kopā</b>	11	120	100	230

1. un 2. daļā ir iekļauti dažāda veida uzdevumi, piemēram, īso un izvērsto atbilžu uzdevumi, informācijas apkopošana un apstrāde, problēmas analīze, risinājuma vai tā daļu modelēšana u. c. Katra no eksāmena daļām var saturēt visu šo veidu uzdevumus. Katra veida uzdevumu skaits un īpatsvars daļā un eksāmenā kopumā nav noteikts. Uzdevuma veida izvēli nosaka atbilstība SR, ko tas pārbauda. Datu modelēšanai un vaicājumu izpildei skolēns drīkst izmantot jebkuru lokāli pieejamo datu bāzu vadības sistēmu, kuru ir apguvis mācību procesā.

3. un 4. daļā ir iekļauti uzdevumi, kuros skolēnam jāizstrādā programmatūra atbilstoši dotajam problēmas aprakstam un uzdevuma nosacījumiem. Skolēns drīkst izmantot jebkuru programmēšanas valodu un vidi, kuru ir apguvis mācību procesā skolā.

Visās daļās ir uzdevumi, kuros tiek vērtēta programmēšanas labās prakses principu ievērošana (sk. 1. pielikumu).



#### 4. EKSĀMENA PIEKĻUVES NOSACĪJUMI

Eksāmenu var kārtot jebkurš skolēns, kas izstrādājis programmatūras produktu un tā dokumentāciju, īstenojot visus programmatūras izstrādes dzīvescikla posmus. Izstrādātās programmatūras dokumentācijas jeb piekļuves materiālu apjomam jābūt līdz 15 (piecpadsmit) A4 formāta lapām, neskaitot titullapu, satura rādītāju un pielikumu/-s. Piekļuves nosacījumi var tikt izpildīti, apgūstot “Programmēšana II. Padziļinātā kursa programmas paraugs vispārējai vidējai izglītībai” 4. tematu “Problēmas analīze, programmatūras specifikācija un darba plānošana” un 5. tematu “Programmatūras izstrāde”. Dokumentācijai ir jāsaturs visas tālāk norādītās nodaļas. Katrā nodaļā dokumentācijas saturam jāatbilst vismaz SLA 2. līmenim, t. i., “Turpina apgūt” (sk. 3. pielikumu), un programmatūras produktam jāatbilst tālāk norādītajām minimālajām prasībām.

Piekļuves materiālus **no 2024. gada 4. marta, bet ne vēlāk kā astoņas nedēļas pirms eksāmena norises dienas, t. i., līdz 2024. gada 24. aprīlim, skolēnam jāaugšupielādē** Valsts pārbaudījumu informācijas sistēmas (turpmāk – VPS) [tīmekļvietnē](#). Kārtība, kādā ir augšupielādējami piekļuves materiāli, kā arī cita skolēniem noderīgā informācija atrodama VPS lietotāju atbalsta [tīmekļvietnē](#).

Piekļuves materiālus izglītības iestādes skolotājs vērtē ballēs atbilstoši izglītības iestādes saistošai izglītojamo mācību sasniegumu vērtēšanas kārtībai un **ne vēlāk kā sešas nedēļas pirms eksāmena norises dienas, t. i., līdz 2024. gada 8. maijam, vērtējumu ievada VPS.**

**Izglītojamais eksāmenu drīkst kārtot, ja vērtējums par piekļuves materiālu nav zemāks par četrām ballēm.** Izglītojamie, kuri eksāmenu kārtu augstskolā, piekļuves materiālus neiesniedz.

Iesniedzamajai dokumentācijai ir jāsaturs šādas nodaļas (sk. arī 3. pielikumu):

**1. Problēmas izpēte un analīze – izpētes metodes izvēle un pamatojums, izpētes procesa apraksts, izpētes datu apkopojums;**

SR: Analizē dažādus ikdienas darba procesus, saskata tajos vai to daļās automatizācijas iespējas. (T.A.2.4.1.)

**2. Programmatūras prasību specifikācija – risinājuma mērķauditorijas izvēle un tās raksturojums, programmatūras produkta un tā funkciju apraksts, programmatūras produkta skice;**

SR: Ikdienas darba procesos vai to daļās saskata automatizācijas iespējas un to, kā pasūtītājs formulē darba uzdevumu programmatūras izstrādātājam. (T.A.2.4.1.)

SR: Sastāda vienkāršotu programmatūras prasību specifikāciju atbilstoši konkrētajam uzdevumam, izvērtējot mērķauditorijas specifiku un vajadzības. (T.A.2.4.4.)

SR: Veic vienkāršotu programmatūras projektēšanu (t. sk. lietotāju saskarnes un vienkāršotu datu modeļa izveidi) atbilstoši programmatūras prasību specifikācijā izvirzītajām funkcionālajām un nefunkcionālajām prasībām. (T.A.2.4.5.)

SR: Izvēlas programmēšanas valodu un programmatūras izstrādes vidi programmatūras izstrādē atbilstoši uzdevuma specifikai, pamato savu izvēli. (T.A.2.4.12.)

**3. Programmatūras izstrādes plāns;**

SR: Salīdzina un izvēlas piemērotāko programmatūras izstrādes modeli konkrētā uzdevuma atrisināšanai, pamato savu izvēli. (T.A.2.4.2.)

SR: Veic vienkāršotu (bez darbietilpības novērtējuma) programmatūras izstrādes plānošanu. (T.A.2.4.5.)

**4. Atklūdošanas un akcepttestēšanas pārskats;**

SR: Veic programmatūras vienību izstrādi un vienībtestēšanu, izstrādājamās programmatūras vienību apvienošanu, integrācijas un akcepttestēšanu atbilstoši izstrādātajai programmatūras prasību specifikācijai un projektējuma aprakstam. (T.A.2.4.6.)

**5. Lietotāja ceļvedis;**

SR: Izstrādā un prezentē izveidotās programmatūras vienkāršoto izvēršanas (t. sk. ieviešanas) plānu, lietotāja ceļvedi un uzturēšanas plānu, ievērojot tās lietotāju mērķauditorijas specifiku. (T.A.2.4.7.)

**6. Piemērotās licences pamatojums;**

SR: Salīdzina atvērto kodu licences un to atšķirības, izmanto un piemēro atbilstošāko no licencēm savam programmatūras projektam. (T.A.3.1.4.)

**7. Programmatūras kods (pievienojams dokumentācijas pielikumā), kas veidots, ievērojot labās prakses principus (sk. 1. pielikumu).**

SR: Veido programmu, ievērojot labās prakses pieredzi tās pieraksta strukturēšanā un komentāru veidošanā. (T.A.2.4.8.)

Iesniedzamajam programmatūras produktam ir jāapmierina šādas minimālās prasības (sk. arī 3. pielikumu):

**1. Ir izmantota datu bāze ar vairākām saistītām tabulām;**

SR: Plāno datu bāzi, t. sk. izveido ER modeli, konkrētā uzdevuma datu apstrādes risinājumam. (T.A.2.3.2.)

SR: Izveido vienkāršu datu apstrādes programmatūru (sistēmu), datu uzglabāšanai, izmantojot paša veidotu datu bāzi ar vairākām saistītām tabulām. (T.A.2.4.17.)

**2. Ir izmantota vismaz viena bibliotēka un/vai API;**

SR: Izmanto programmēšanas valodas un tās bibliotēku dokumentāciju un palīdzības sistēmu, lai patstāvīgi apgūtu citas to piedāvātās iespējas, kas nepieciešamas konkrētās programmatūras izstrādei. (T.A.2.4.10.)

SR: Meklē un pievieno atvērtā koda bibliotēkas, kā arī izmanto API specializētu funkciju veikšanai sava programmēšanas projekta īstenošanai. (T.A.2.4.11.)

SR: Izvēlas un lieto atbilstošas programmēšanas valodas konstrukcijas, datu tipus un dažādas bibliotēkas, veidojot programmas doto uzdevumu un problēmu risinājumam. (T.A.2.4.13.)

**3. Ir izmantota vismaz viena no datu struktūrām;**

SR: Izmanto dažādas datu struktūras (piemēram, masīvs, saraksts, kopa, ieraksts, steks, rinda, grafs, koks u. c.) un ar tiem saistītos pamatalgoritmus. (T.A.2.4.14.)

SR: Veido dotā uzdevuma (problēmas) risinājumu, izmantojot gatavus algoritmus un/vai pielāgojot vai kombinējot tos, un/vai izstrādājot jaunus algoritmus. Izprot un skaidro dažādu algoritmu darbību, pielāgo tos dažādām nestandarta situācijām, ja nepieciešams, veidojot jaunas datu struktūras. (T.A.2.4.19.)

**4. Ir realizēta lietotāja piekļuves un datu aizsardzība.**

SR: Izmanto kriptogrāfijas metodes konkrētā uzdevuma risinājumā. (T.A.3.1.2.)

## 5. NEPIECIEŠAMO RESURSU NODROŠINĀJUMS

Eksāmens veicams, izmantojot izglītības iestādes datoru.

Pie izglītojamajiem un personām, kuras piedalās eksāmena nodrošināšanā, no brīža, kad viņiem ir pieejams eksāmena materiāls, līdz eksāmena norises beigām nedrīkst atrasties ierīces (planšetdators, piezīmjdators, viedtālrunis, viedpulkstenis u. c. saziņas un informācijas apmaiņas līdzekļi), kuras nav paredzētas eksāmena norises darbību laikos.

Eksāmena **1., 3. un 4. daļas praktiskajos uzdevumos** drīkst izmantot operētājsistēmas pamatprogrammas un biroja programmatūru (piemēram, kalkulatoru, datņu pārlūku, izklājlapu redaktoru), interneta pārlūkprogrammu (tikai atļautajām darbībām, sk. 7. nodaļu), koda rakstīšanas un programmatūras izstrādes vides un/vai to atsevišķos komponentus.

## 6. VĒRTĒŠANAS KĀRTĪBA UN KRITĒRIJI

### 6.1. Vērtēšanas kārtība

Katrā uzdevumā ir norādīts maksimālais iegūstamo punktu skaits. Eksāmena vērtētājam ir pieejami kritēriji, pēc kuriem nosaka punktu skaitu, ko skolēns iegūvis. Skolēna rezultātus eksāmenā – iegūto punktu summu visā darbā un iegūto punktu summu katrā daļā – izsaka procentuālā novērtējumā. **Vidēji 20 % eksāmenā iekļauto testelementu reprezentē minimālo prasību kopumu – katra eksāmena satura moduļa izpildi atbilstoši 1. un 2. līmenim SOLO taksonomijā** (piemēram, atpazīst pamatalgoritmus, nosauc piemērotāko izpētes metodi, izpilda vienkāršas darbības, nodrošina datu ievadizvadi, kombinē zināšanas un prasmes standarta situācijās utt.).

Atbilstoši Ministru kabineta 2019. gada 3. septembra noteikumiem Nr. 416 “Noteikumi par valsts vispārējās vidējās izglītības standartu un vispārējās vidējās izglītības programmu paraugiem” 25.<sup>12.</sup> punktam **eksāmenā vērtējums nav iegūts, ja darba kopvērtējums 2023./2024. mācību gadā ir mazāks nekā 15 %.**

### 6.2. Vērtēšanas kritēriji

Skolēnu sniegumu eksāmenā vērtē pēc kritērijiem, kas var būt izteikti kā katram punktam atbilstošu atbilžu, darbību, rezultāta apraksts. Programmēšanas labās prakses principu ievērošana tiek vērtēta pēc snieguma līmeņu apraksta (turpmāk – SLA), katram līmenim piešķirot noteiktu punktu skaitu (sk. 1. pielikumu).

Eksāmenā biežāk lietoto rīcības vārdu skaidrojums (2. pielikums) ir iekļauts, lai veidotu vienotu skolotāju un skolēnu izpratni par šo vārdu nozīmi un tai atbilstošu skolēna sniegumu mācību procesā un eksāmenā.

## **7. PALĪGLĪDZEKĻI, KURUS ATĻAUTS IZMANTOT EKSĀMENA LAIKĀ**

1. Izvēlēto programmēšanas valodu dokumentācijas.
2. Izvēlētā, lokāli instalētā, datu bāzu vadības sistēma (piemēram, *DB Browser for SQLite*, *Microsoft SQL Server* u.c.).
3. Izvēlētās integrētās izstrādes vides (piemēram, *Visual Studio*, *PyCharm*, *Eclipse*, *NetBeans*, *IntelliJ* u. c.) un koda redaktori (piemēram, *Visual Studio Code*, *Notepad*, *Notepad++* u. c.).
4. Tīmekļa vietnes papildinformācijas meklēšanai un kļūdu labojumiem <https://www.w3schools.com> (bez autorizācijas).

**PIELIKUMI**



## Programmēšanas labās prakses principi

Daži programmēšanas labās prakses principu piemēri ir šādi:

1. Katru atsevišķu priekšrakstu raksta jaunā rindā;
2. Koda loģiskās daļas (piemēram, zarošanos, ciklu, masīvu) savstarpēji atdala ar tukšu rindu;
3. Izmanto atkāpes, lai vizualizētu priekšrakstu vai struktūru iekļaušanu citās struktūrās;
4. Izvairās no garām koda rindām, izmanto pārnesi jaunā rindā, atvieglotot koda lasīšanu;
5. Mainīgo, funkciju u. c. nosaukumus veido jēgpilnus, atvieglotot koda uztveri (piemēram, perimetru apzīmējot nevis ar “p”, bet “perim” vai “perimets”);
6. Ar komentāriem skaidro programmatūras koda loģiskās daļas (piemēram, zarošanos, ciklu, masīvu), to lomu programmatūrā (piemēram “datu izvade”).

<b>Līmenis</b> <b>Kritērijs</b>	<b>Sācis apgūt</b>	<b>Turpina apgūt</b>	<b>Apguvis</b>	<b>Apguvis padziļināti</b>
<b>Veido programmatūru, ievērojot labās prakses principus tās pieraksta strukturēšanā un komentāru veidošanā</b>	Lielākajā daļā programmatūras koda lieto labās prakses principus, bet dara to nekonsekventi vai daļēji korekti.	Labās prakses principus programmatūras kodā lieto kopumā korekti un konsekventi, pieļaujot dažas neprecizitātes.	Labās prakses principus programmatūras kodā lieto korekti un konsekventi.	Labās prakses principus programmatūras kodā lieto vienmēr korekti un konsekventi, patstāvīgi piedāvā, ievieš un izmanto līdzīgos principus.

## Rīcības vārdu skaidrojums

Eksāmenā biežāk lietoto vai mācību procesā nereti dažādi interpretēto rīcības vārdu skaidrojums (sk. 5. tabulu) ir iekļauts, lai veidotu vienotu skolotāju un skolēnu izpratni par šo vārdu nozīmi un tai atbilstošu skolēna sniegumu mācību procesā un eksāmenā.

### 5. tabula

#### Biežāk lietotie rīcības vārdi un to skaidrojums

Rīcības vārds	Skaidrojums
<b>Analizē</b>	Sīki, pamatīgi pēta, raksturo, piemēram, algoritmu, problēmas aprakstu.
<b>Apvieno</b>	Izveido vienu veselu (no atsevišķām vienībām, daļām, piemēram, apvieno programmatūras daļas).
<b>Atpazīst</b>	Pazīst, identificē (kādu/ko pēc noteiktām pazīmēm, parasti starp līdzīgiem, piemēram, OOP darbības principus).
<b>Formulē</b>	Izsaka (domu, atzinumu, lēmumu, jautājumu), piemēram, kādu programmatūras produktu nepieciešams izstrādāt.
<b>Identificē</b>	Nosaka, pazīst; konstatē (ko/kādu pēc kādām raksturīgām pazīmēm, piemēram, lomu sadalījumu projekta komandā).
<b>Ievēro</b>	Rīkojas saskaņā (ar ieteikumiem, norādījumiem u. tml.); iegaumē un izpilda (noteikumus, prasības u. tml.). Piemēram, labās prakses principus, rakstot programmatūras kodu.
<b>Integrē</b>	Iekļauj kādā vienībā; papildina, pilnveido, izveido par veselu vienību (piemēram, programmatūras daļas vienā veselumā).
<b>Izmanto</b>	Lieto (ko) par pamatu, materiālu (piemēram, kā noteikta gatavošanai, izveidošanai), piemēram, programmēšanas valodas dokumentāciju.
<b>Izprot</b>	Pilnīgi izzina un saprot, piemēram, algoritma darbību.
<b>Izveido/Izstrādā</b>	Ar mērķtiecīgu darbību panāk vai organizē, ka, piemēram, datubāze (vai kas cits) iegūst vēlamo veidu un atbilst noteiktām prasībām; rada prasīto.
<b>Izvēlas</b>	Iegūst lietošanai, atrod, izraugās izmantošanai no kāda kopuma (ko piemērotu, atbilstošu, piemēram, programmatūras izstrādes modeli savam projektam atbilstoši problēmai, nosacījumiem u. tml.).
<b>Izvērtē</b>	Novērtē kādu kopumu, parasti pa tā sastāvdaļām, detaļām; katru faktu, notikumu u. tml. noteiktā kopsakarībā (piemēram, mērķauditorijas specifiku un vajadzības).
<b>Kombinē</b>	Veido noteiktā veselumā (no atsevišķām sastāvdaļām), piemēram, vairākas algoritma daļas, savienojot vienā veselumā.

<b>Rīcības vārds</b>	<b>Skaidrojums</b>
<b>Konfigurē</b>	Pielāgo sistēmu (piemēram, datortīklu), uzstādot aparāturu un programmatūru (piemēram, maršrutētāja); noteiktā veidā uzstāda.
<b>Lieto</b>	Rīkojas (ar materiālu, koda fragmentu) mērķtiecīgi, pārdomāti – tā, lai gūtu labumu; izmanto (piemēram, lieto programmsaskarni).
<b>Meklē</b>	Cenšas (skatoties u. tml.) dabūt (ko vajadzīgu, nepieciešamu u. tml., piemēram, atvērtā koda bibliotēkas).
<b>Pamato</b>	Mīn faktus, cēloņus, loģiskus slēdzienus u. tml., kas pierāda (kā) patiesumu, nepieciešamību (piemēram, izvēli lietot konkrētu izpētes metodi lietotāju vajadzību noskaidrošanai).
<b>Pārbauda</b>	Pārlicinās, vai (kas) atrodas noteiktā stāvoklī, kārtībā (piemēram, programmatūras veiktspēju, programmatūras kodu), arī – vai kas ir noticis, izdarīts (piemēram, iegūts pareizais rezultāts, izpildot programmu).
<b>Pielāgo</b>	Veido, izveido (ko, piemēram, algoritmu) atbilstoši izmantošanas mērķim, apstākļiem.
<b>Piemēro</b>	Padara atbilstošu (piemēram, apstākļiem, iespējām programmatūras licenci).
<b>Pievieno</b>	Pieliek (pie kā, kam klāt, piemēram, atvērtā koda bibliotēkas).
<b>Plāno</b>	Veido (kāda objekta, piemēram, datubāzes) plānu, skici, shēmu. Iepriekš sadala (laiku) pa posmiem (kā veikšanai, piemēram, projekta īstenošanai).
<b>Raksturo</b>	Nosaka, apraksta, vērtē (kā) raksturīgās īpašības, pazīmes. Piemēram, mašīnmācīšanās algoritmus.
<b>Salīdzina</b>	Pēta, analizē, novēro u. tml. (divus vai vairākus priekšmetus, blokshēmas u. tml.), lai noteiktu (to) kopīgās vai atšķirīgās īpašības, pazīmes.
<b>Saskata</b>	Skatoties, pilnīgi precīzi uztver (ko, piemēram, automatizācijas iespējas), atšķir, pazīst (ko); skatoties atklāj, konstatē (ko), secina (ko).
<b>Sastāda</b>	Izveido (piemēram, programmatūras prasību specifiku), apvienojot materiālus; uzraksta noteiktā formā, izmantojot kādus materiālus.
<b>Skaidro</b>	Stāstot, rakstot, arī rādot panāk, ka kādam (kas) kļūst zināms, saprotams (piemēram, programmatūras koda darbības principi).
<b>Veic</b>	Ar savu darbību, rīcību panāk, ka īstenojas (piemēram, kāda norise), tiek sasniegts (kāds mērķis). Piemēram, veic akcepttestēšanu.

Rīcības vārdu skaidrojumi ar uzdevumu piemēriem ir minēti 6. tabulā. Šīs tabulas virsrakstu skaidrojums:

- SR jeb sasniedzamais rezultāts – atbilstoši standartam;
- SOLO jeb novēroto mācīšanās rezultātu taksonomija – maksimālais SR pārbaudes dziļums atbilstoši SOLO taksonomijai, eksāmenā SR var pārbaudīt zemākā, bet ne augstākā SOLO līmenī;
- Rīcības vārdi – SR iekļautie darbības (rīcības) vārdi;
- Uzdevumu veidu piemēri – rīcības vārdu interpretācija dažādu veidu uzdevumu piemēros (piemēru saraksts nav pilnīgs);
- Komentāri – papildu informācija uzdevumu veidu piemēriem.

**6. tabula**

**Biežāk lietotie rīcības vārdi un to skaidrojums**

<b>SR</b>	<b>SOLO</b>	<b>Rīcības vārdi</b>	<b>Uzdevumu veidu piemēri</b>	<b>Komentāri</b>
T.A.2.3.2.	3	Plāno	<ul style="list-style-type: none"><li>• Uzzīmē uz papīra vai izmantojot lietotni datu bāzes shēmu ar vairākām saistītām tabulām.</li></ul>	

SR	SOLO	Rīcības vārdi	Uzdevumu veidu piemēri	Komentāri
T.A.2.4.1.	3	Analizē un saskata	<ul style="list-style-type: none"> <li>Iepazīstas ar problēmas aprakstu teksta formātā un uzskaita dažādas procesu automatizācijas iespējas.</li> <li>Ir dots problēmas apraksts un varianti automatizācijas risinājumiem – identificē, kuri no tiem atbilst dotajai situācijai un kuri neatbilst.</li> </ul>	Dizaina domāšanas principi un soļi: izpēte, problēmas definēšana, ideju ģenerēšana u. c.
		Formulē	<ul style="list-style-type: none"> <li>Balstoties uz dotu problēmas aprakstu, formulē, kādu programmatūras produktu nepieciešams izstrādāt.</li> <li>Uzskaita un apraksta tehniskās prasības programmatūras produkta izstrādei.</li> </ul>	
T.A.2.4.2.	3	Salīdzina	<ul style="list-style-type: none"> <li>Atpazīst un raksturo vienu vai vairākus programmatūras izstrādes modeļus.</li> <li>Uzskaita ieguvumus, riskus konkrētu programmatūras izstrādes modeļu izmantošanā un salīdzina divus vai vairākus programmatūras izstrādes modeļus.</li> </ul>	
		Izvēlas un pamato	<ul style="list-style-type: none"> <li>Ņemot vērā problēmas aprakstu, izvēlas un pamato, kuru programmatūras izstrādes modeli izmantot.</li> <li>Plāno projekta izstrādes procesu, iekļaujot tajā programmatūras izstrādes modeli un paskaidrojumu par tā izvēli.</li> </ul>	

SR	SOLO	Rīcības vārdi	Uzdevumu veidu piemēri	Komentāri
T.A.2.4.4.	4	Izvērtē	<ul style="list-style-type: none"> <li>Definē un raksturo mērķauditoriju atbilstoši dotajam uzdevumam.</li> <li>Atbilstoši dotajai problēmai identificē un apkopo informāciju par mērķauditorijas specifiku un vajadzībām.</li> <li>Pielāgo programmatūras produkta ideju dažādām mērķauditorijām.</li> </ul>	
		Izveido [programmatūras prasību specifikāciju]	<ul style="list-style-type: none"> <li>Balstoties uz doto problēmas aprakstu, izstrādā programmatūras prasību specifikāciju vai kādas tās daļas.</li> <li>Papildina iesākto programmatūras prasību specifikāciju.</li> </ul>	
		Veic [akcepttestēšanu]	<ul style="list-style-type: none"> <li>Veic akcepttestēšanu atbilstoši izstrādātajām programmatūras prasībām.</li> <li>Fiksē atbilstības un neatbilstības.</li> </ul>	
T.A.2.4.8.	4	Ievēro [labās prakses principus]	<ul style="list-style-type: none"> <li>Labo programrisinājuma pirmkodu atbilstoši labās prakses principiem (sk. 1. pielikumu).</li> </ul>	

SR	SOLO	Rīcības vārdi	Uzdevumu veidu piemēri	Komentāri
T.A.2.4.10.	4	Analizē un izmanto [piedāvātās iespējas]	<ul style="list-style-type: none"> <li>Izstrādājot programmatūru, jēgpilni izmanto esošo bibliotēku piedāvātās iespējas.</li> <li>Lieto esošo klašu konstruktorus un to metodes, tādējādi saīsinot esošo programmatūras kodu un uzlabojot izstrādājamās programmatūras darba efektivitāti.</li> <li>Analizē doto bibliotēku dokumentāciju un palīdzības sistēmu informāciju, lai atrastu un pielietotu atbilstošu bibliotēku problēmas risināšanā.</li> <li>Izmanto programmēšanas valodas dokumentāciju, lai apgūtu programmēšanas valodas iespējas dotās problēmas risināšanā (t. sk. sintakses pārbaudē).</li> </ul>	
T.A.2.4.11.	4	Meklē	<ul style="list-style-type: none"> <li>Atrod atvērtā koda bibliotēkas, kas veic nepieciešamās funkcijas, un pamato savu izvēli.</li> </ul>	Piemēram, pamato savu atbildi ar saiti un/vai citātu no dokumentācijas.
		Pievieno	<ul style="list-style-type: none"> <li>Pievieno paša atrastās un/vai dotās bibliotēkas.</li> </ul>	
		Lieto	<ul style="list-style-type: none"> <li>Pielieto bibliotēkas un/vai API dotā uzdevuma risinājumam.</li> </ul>	

SR	SOLO	Rīcības vārdi	Uzdevumu veidu piemēri	Komentāri
T.A.2.4.14.	3	Izmanto	<ul style="list-style-type: none"> <li>Izveido risinājumu dotajam uzdevumam, izmantojot dažādas datu struktūras.</li> </ul>	
T.A.2.4.15.	4	Skaidro [OOP pamatprincipus]	<ul style="list-style-type: none"> <li>Skaidro klašu, īpašību un metožu izveides un pielietošanas iespējas, piemēram, pēc dotā problēmas definējuma uz papīra vai lietotnē uzraksta pseidokodu ar piemēriem.</li> </ul>	
		Veido [programmas]	<ul style="list-style-type: none"> <li>Veido programmatūras produktu, balstoties uz OOP pamatprincipiem – izveido un pielieto klases un to īpašības, metodes un konstruktorus.</li> </ul>	
T.A.2.4.17.	4	Izveido	<ul style="list-style-type: none"> <li>Izveido programmatūras produktu.</li> </ul>	



### Projekta izstrādes dokumentācijas snieguma līmeņu apraksts

Šis SLA var tikt izmantots, izstrādājot projektus vai to daļas optimālajā un padziļinātajā līmenī, veicot pašnovērtējumu, summatīvo vai formatīvo vērtēšanu mācību procesā, kā arī eksāmena piekļuves nosacījumu izpildes vērtēšanai.

Snieguma līmeņu aprakstu programmatūras koda (pievienojams dokumentācijas pielikumā), kas veidots, ievērojot labās prakses principus, veidošanas posmam skat. atsevišķi – 4. pielikumā.

<b>Līmenis Kritērijs</b>	<b>Sācis apgūt</b>	<b>Turpina apgūt</b>	<b>Apguvis</b>	<b>Apguvis padziļināti</b>
<b>1. Problēmas izpētes metodes izvēle un pamatojums, izpētes procesa apraksts, izpētes datu procesa apraksts, izpētes datu apkopojums</b>	Izpētes procesa soļi ir sajauktā secībā un/vai neveido loģisku procesu (iztrūkst kāds būtisks posms, nav sākuma vai noslēguma u. tml.).	Aprakstā ir mazāk par pieciem procesa soļiem, tie ir loģiskā secībā un atspoguļo izpētes procesu no sākuma līdz galam, piemēram: sagatavošanās, norise un secinājumu izdarīšana.	Aprakstā iekļautā izpētes procesa gaita ir loģiska, detalizēta un sastāv no vismaz pieciem soļiem, piemēram: sagatavošanās, norise, rezultātu apkopošana, izvērtēšana un secinājumu izdarīšana.	Precīzi aprakstīta problēmas tēma sasaistē ar problēmas izpētes mērķi, kas formulēti konkrēti un saprotami.  Izmantoto problēmas izpētes metožu pielietošana ir pamatota un jēgpilna, izpētes dati ir daudzveidīgi un sniedz atbildes uz izpētes jautājumiem.
	<i>1 punkts</i>	<i>2 punkti</i>	<i>3 punkti</i>	<i>4 punkti</i>

<b>Līmenis</b> <b>Kritērijs</b>	<b>Sācis apgūt</b>	<b>Turpina apgūt</b>	<b>Apguvis</b>	<b>Apguvis padziļināti</b>
<b>2. Programmatūras risinājuma mērķauditorijas izvēle un tās raksturojums, programmatūras izstrādes projekta risinājuma un tā ievadizvades datu apraksts</b>	Pielikumā pievienotie ievadizvades dati daļēji atbilst pētāmajai problēmai.	Pielikumā pievienotie ievadizvades dati atbilst pētāmajai problēmai. Norādīts ievadizvades datu avots.	Pielikumā pievienotie ievadizvades dati atbilst pētāmajai problēmai. Norādīts ievadizvades datu avots un metožu nosaukumi ar korekto deklarāciju.	Pielikumā pievienotie ievadizvades dati atbilst pētāmajai problēmai. Norādīts ievadizvades datu avots un metožu nosaukumi ar korekto deklarāciju, datu tipi, mērvienības, iespējamās vērtības. Norādīts, kā tiek iegūts (aprēķināts, izvadīts, formatēts utt.) rezultāts.
	<b>1 – 2 punkti</b>	<b>3 – 4 punkti</b>	<b>5 – 6 punkti</b>	<b>7 – 8 punkti</b>
<b>3. Programmatūras izstrādes plāns ar pamatojumu</b>	Iesniegts vispārīgs sagatavošanās posma, intervēšanas norises vai datu apstrādes apraksts.	Iesniegts sagatavošanās posma, intervēšanas norises vai datu apstrādes apraksts. Visu posmu rezultāti ir apkopoti.	Iesniegts sagatavošanās posma, intervēšanas norises vai datu apstrādes apraksts. Visu posmu rezultāti ir apkopoti un kritiski izvērtēti.	Iesniegts sagatavošanās posma, intervēšanas norises vai datu apstrādes apraksts. Visu posmu rezultāti ir apkopoti un kritiski izvērtēti. No rezultātiem ir izdarīti un noformulēti secinājumi.
	<b>1 punkts</b>	<b>2 punkti</b>	<b>3 punkti</b>	<b>4 punkti</b>

<b>Kritērijs</b> / <b>Līmenis</b>	<b>Sācis apgūt</b>	<b>Turpina apgūt</b>	<b>Apguvis</b>	<b>Apguvis padziļināti</b>
<b>4. Atklūdošanas un akcepttestēšanas pārskats</b>	<p>Testēšanas plāns paredz pārbaudīt tikai atsevišķas sistēmas daļas.</p> <p>Akcepttestēšanas pārskats ir vispārīgs un satur virspusēju programmatūras prasību specifikācijā esošo prasību pārbaudi no lietotāja skatupunkta.</p>	<p>Testēšanas plāns ir nepilnīgs, tajā trūkst informācijas par atsevišķu sistēmas daļu testēšanu.</p> <p>Ne visas specifikācijā minētās programmatūras produkta funkcijas tiek pārbaudītas.</p> <p>Akcepttestēšanas pārskats ir vispārīgs un satur lielāko daļu programmatūras prasību specifikācijā esošo prasību pārbaudi no lietotāja skatupunkta.</p>	<p>Testēšanas plānā noteikts, kuras sistēmas daļas un kādā veidā jātestē.</p> <p>Nosaka nepieciešamo rezultātu atbilstoši specifikācijā esošajam programmatūras produkta funkciju aprakstam.</p> <p>Akcepttestēšanas pārskatā atspoguļota sistēmas darbības pārbaude no lietotāja skatupunkta atbilstoši specifikācijai.</p>	<p>Testēšanas plānā noteikts, kuras sistēmas daļas un kādā veidā jātestē.</p> <p>Nosaka nepieciešamo rezultātu atbilstoši specifikācijā esošajam programmatūras produkta funkciju aprakstam.</p> <p>Akcepttestēšanas pārskatā atspoguļota sistēmas darbības pārbaude no vairākiem lietotāju grupu skatupunktiem atbilstoši specifikācijai.</p> <p>Testēšanai ir izmantots digitāls rīks un/vai bibliotēka.</p>
		<i>1 – 2 punkti</i>	<i>3 – 4 punkti</i>	<i>5 – 6 punkti</i>

<b>Kritērijs</b> / <b>Līmenis</b>	<b>Sācis apgūt</b>	<b>Turpina apgūt</b>	<b>Apguvis</b>	<b>Apguvis padziļināti</b>
<b>5. Lietotāja ceļvedis</b>	Izmantota iespējami vienkārša un saprotama valoda. Pievienota vizuāla informācija.	Izmantota iespējami vienkārša un saprotama valoda. Pievienota vizuāla informācija. Loģiska dokumenta struktūra. Viegli saprotamas sadaļas.	Izmantota iespējami vienkārša un saprotama valoda. Pievienota vizuāla informācija. Loģiska dokumenta struktūra. Viegli saprotamas sadaļas. Iespēja viegli atrast nepieciešamo informāciju. Iekļauta tikai nepieciešamā informācija.	Izmantota iespējami vienkārša un saprotama valoda. Pievienota vizuāla informācija. Loģiska dokumenta struktūra. Viegli saprotamas sadaļas. Iespēja viegli atrast nepieciešamo informāciju. Iekļauta tikai nepieciešamā informācija. Ceļveža izstrādei un publicēšanas tiešsaistē ir izmantots digitāls rīks.
	<i>1 punkts</i>	<i>2 punkti</i>	<i>3 punkti</i>	<i>4 punkti</i>

<b>Līmenis</b> <b>Kritērijs</b>	<b>Sācis apgūt</b>	<b>Turpina apgūt</b>	<b>Apguvis</b>	<b>Apguvis padziļināti</b>
<b>6. Produkta licence</b>	Norādīta atbilstošā licence.	Norādīta atbilstošā licence. Izvēle ir pamatota.	Norādīta atbilstošā licence. Izvēle ir pamatota ar dokumentācijā ietverto informāciju un/vai izmantotajām tehnoloģijām, bibliotēkām utt.	Ir pieminēti vairāki licenču varianti. Ir veikta salīdzinoša analīze ar katra varianta priekšrocībām un trūkumiem. Rezultātā, balstoties uz analīzes secinājumiem, dokumentācijā ietverto informāciju un/vai izmantotajām tehnoloģijām, bibliotēkām utt., ir izvēlēts viens atbilstošākais variants.
	<i>1 punkts</i>	<i>2 punkti</i>	<i>3 punkti</i>	<i>4 punkti</i>

### Snieguma līmeņu apraksts programmatūras koda veidošanas posmam

<b>Līmenis</b> <b>Kritērijs</b>	<b>Sācis apgūt</b>	<b>Turpina apgūt</b>	<b>Apguvis</b>	<b>Apguvis padziļināti</b>
<b>1. Datu bāzes izveide un izmantošana</b>	Plāno datu bāzi, t. sk. izveido ER modeli, konkrētā uzdevuma datu apstrādes risinājumam.	Plāno datu bāzi, t. sk. izveido ER modeli, konkrētā uzdevuma datu apstrādes risinājumam.  Izmanto kriptogrāfijas metodes konkrētā uzdevuma risinājumā.	Izveido vienkāršu datu apstrādes programmatūru datu nolasīšanai, izmantojot paša veidoto datu bāzi ar vienu tabulu un izmantojot kriptogrāfijas metodes datu aizsardzībai.	Izveido vienkāršu datu apstrādes programmatūru datu nolasīšanai un saglabāšanai, izmantojot paša veidoto datu bāzi ar vairākām saistītām tabulām un izmantojot kriptogrāfijas metodes datu aizsardzībai.
	<i>1 punkts</i>	<i>2 punkti</i>	<i>3 punkti</i>	<i>4 punkti</i>
<b>2. API izmantošana</b>	Ir kaut viena ideja par iespējamiem API izmantošanas variantiem, un ir sākta viena API integrēšana programmatūrā.	Ir vairākas idejas par iespējamiem API izmantošanas variantiem, no kuriem ir izvēlēts visatbilstošākais, un ir sākta tā integrēšana programmatūrā.	Ir kritiski izvēlēts un prasmīgi integrēts pedagoga ieteiktais API, kas korekti darbojas, bet tas pilnībā neatbilst programmatūras uzdevumiem un mērķiem.	Ir kritiski izvēlēts un prasmīgi integrēts patstāvīgi atrastais API, kas korekti darbojas un pilnīgi atbilst programmatūras uzdevumiem un mērķiem.
	<i>1 punkts</i>	<i>2 punkti</i>	<i>3 punkti</i>	<i>4 punkti</i>

<b>Kritērijs</b> / <b>Līmenis</b>	<b>Sācis apgūt</b>	<b>Turpina apgūt</b>	<b>Apguvis</b>	<b>Apguvis padziļināti</b>
<b>3. Datu ievadizvades organizēšana</b>	Visi dati tiek tikai ievadīti.	Dati tiek gan ievadīti, gan izvadīti.	Dati vairākumā tiek ne tikai ievadīti un izvadīti, bet arī saglabāti un apstrādāti.	Visi dati tiek ievadīti, saglabāti, apstrādāti un izvadīti.  Datu tiek ielādēti un saglabāti ārējos datu avotos, piemēram, datnēs, tīmekļa serveros utt.
	<i>1 punkts</i>	<i>2 punkti</i>	<i>3 punkti</i>	<i>4 punkti</i>
<b>4. OOP principu izmantošana</b>	Ir definēta viena vai vairākas klases, bet nav realizēta mantošana: nav bāzes un atvasināto klašu.	Ir definēta viena bāzes klase un viena atvasinātā klase.	Ir definēta vismaz viena bāzes klase un 2 atvasinātās klases.	Ir definēta vismaz viena bāzes klase un 2 atvasinātās klases.  Ir ievēroti OOP labās prakses principi.
	<i>1 punkts</i>	<i>2 punkti</i>	<i>3 punkti</i>	<i>4 punkti</i>