
# Padziļinātais kurss Programmēšana II augstākajā mācību satura apguves līmenī

## Valsts pārbaudes darba programma

Valsts izglītības satura centrs | ESF projekts Nr. 8.3.1.1/16/I/002
Kompetenču pieeja mācību saturā

SKOLA 2030

# Padziļinātais kurss Programmēšana II augstākajā mācību satura apguves līmenī

## Valsts pārbaudes darba programma

Valsts pārbaudes darba paraugs ir izstrādāts Eiropas Sociālā fonda projektā "Kompetenču pieeja mācību saturā" (turpmāk – Projekts).

Valsts pārbaudes darbu satura, programmu un paraugu izstrādi Projektā vadīja **Pāvels Pestovs.**

Valsts pārbaudes darba programmas izstrādi un sagatavošanu publicēšanai vadīja **Kaspars Antonevičs.**

Valsts pārbaudes darba programmu izstrādāja **Ance Kancere** un **Edgars Kupčs.**

Valsts pārbaudes darba programmas mācību satura recenzents: **Sergejs Zembkovskis.**

Valsts pārbaudes darba programmas zinātniskais recenzents: **Kristaps Ozols.**

***Projekts izsaka pateicību visām Latvijas izglītības iestādēm, kas piedalījās valsts pārbaudes darba parauga aprobācijā.***

# Saturs

# Ievads

Valsts pārbaudes darba programma veidota kā teorētiskais ietvars 2022./2023., kā arī nākamo mācību gadu valsts pārbaudes darbu izstrādei, kas nodrošinātu to salīdzināmību, grūtības pakāpes nemainību un pēctecību.

Programmā skaidrots vērtēšanas saturs, pārbaudes darba uzbūve, vērtēšanas kārtība un kritēriji, piekļuves nosacījumi valsts pārbaudes darbam, kā arī izmantojamie palīglīdzekļi.

# 1. Valsts pārbaudes darba mērķis

Valsts pārbaudes darba 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), lai sertificētu apgūto, kā arī nepieciešamības gadījumā izmantotu iegūto informāciju, lai atlasītu skolēnus pēc vienotiem kritērijiem.

Iegūtā informācija sniedz iespēju izvērtēt skolēnu sniegumu un mācību saturu, izstrādāt metodiskos ieteikumus un plānot profesionālo pilnveidi izglītības iestādes, dibinātāja un valsts līmenī.

Valsts pārbaudes darba 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 līmeņa valsts pārbaudes darba (turpmāk – VPD) vērtēšanas saturu raksturo trīs kategorijas:

1) sasniedzamo rezultātu veids un grupa;
2) satura modulis;
3) izziņas darbības līmenis.

Tas nozīmē, ka katru VPD testelementu[1] raksturo noteikts SR veids un grupa, satura modulis un izziņas darbības līmenis.

## 2.1. Sasniedzamo rezultātu veids un grupa

Skolēnam plānoti triju veidu SR:

1) zināšanas un izpratne;
2) prasmju grupas;
3) zināšanu, izpratnes, prasmju un ieradumu kombinācijas.

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

Zināšanu un izpratnes pārbaudei VPD iekļauti uzdevumi, kuros skolēni:

1) atpazīst pamatalgoritmus un izprot to darbības principus, spēj lasīt un izprast blokshēmās un pseidokodā rakstīto. Izprot kriptogrāfijas metožu nepieciešamību, datortīkla uzbūvi u. c.;
2) salīdzina programmatūras izstrādes modeļus, skaidro programmēšanas jēdzienus, raksturo mašīnmācīšanās izmantošanas iespējas u. c.

[1] Testelements ir uzdevums vai uzdevuma daļa, kas veidota, lai pēc kritērijiem vērtētu kādu konkrētu skolēnu darbības aspektu.

Prasmju apguvi raksturo trīs SR grupas:

1) izstrādā programmatūras prasību specifikāciju, programmatūru, izvēršanas plānu u. c., veic atkļūdošanu. Atbilstoši standartā noteiktajam mācību saturam uzdevumu īpatsvars šo prasmju pārbaudei VPD ir vislielākais;
2) lieto prasmes darbā ar informāciju, piemēram, izstrādājot programmatūru, jēgpilni izmanto esošo bibliotēku piedāvātās iespējas, izmanto programmēšanas valodas dokumentāciju, lai apgūtu programmēšanas valodas iespējas dotās problēmas risināšanā (t. sk. sintakses pārbaudē) u. c.
3) lieto labās prakses principus (sk. 1. pielikumu). Vērtējumu par labās prakses principu lietošanu skolēns iegūst eksāmena uzdevumos, kuru vērtēšanas kritērijos norādīts par šo prasmju pārbaudi.

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 dzīves cikla posmi: problēmas analīze, programmatūras specifikācija un darba plānošana, programmatūras produkta izstrāde, akcepttestēšana, atkļūdošana, ieviešana un uzturēšana. Var tikt pārbaudīta skolēnu spēja saskatīt algoritmu optimizācijas iespējas, izvērtēt drošības riskus.

**1.tabula.** Sasniedzamo rezultātu veidi, grupas un to īpatsvars VPD.

| Sasniedzamo rezultātu veids un grupa | | Īpatsvars (%) |
|---|---|---|
| **Zināšanas un izpratne** | Atpazīst pamatalgoritmus un izprot to darbības principus, spēj lasīt un izprast blokshēmās un pseidokodā rakstīto. Izprot kriptogrāfijas metožu nepieciešamību, datortīkla uzbūvi u. c. | 10 ± 5 |
| | Salīdzina programmatūras izstrādes modeļus, skaidro programmēšanas jēdzienus, raksturo mašīnmācīšanās izmantošanas iespējas u. c. | |
| **Prasmju grupas** | Izstrādā programmatūras prasību specifikāciju, programmatūru, izvēršanas plānu u. c., veic atkļūdošanu. | 57 ± 5 |
| | Lieto prasmes darbā ar informāciju. | 8 ± 3 |
| | Lieto programmēšanas labās prakses principus. | 3 ± 1 |
| **Zināšanu, izpratnes, prasmju un ieradumu kombinācijas** | Kombinē vairākus programmatūras dzīves cikla posmus: problēmas analīzi, programmatūras specifikāciju un darba plānošanu, programmatūras produkta izstrādi, akcepttestēšanu, atkļūdošanu, ieviešanu un uzturēšanu. Saskata algoritmu optimizācijas iespējas, izvērtē drošības riskus. | 16 ± 4 |

## 2.2. Satura moduļi

Satura moduļi VPD strukturēti atbilstoši pieejai kursa "Programmēšana II" programmas paraugā. Satura moduļu īpatsvars VPD darbā (sk. 2. tabulu) ir proporcionāls to apguvei noteiktajam stundu skaitam atbilstošajos satura tematos programmas paraugā.

**2. tabula.** Satura moduļi un to īpatsvars VPD.

| Satura modulis | Īpatsvars (%) |
|---|---|
| Objektorientētā programmēšana un ārējās bibliotēkas | 33 ± 5 |
| Datortīkls un datubāze | 15 ± 5 |
| Datu struktūras, programmsaskarne (API) un mašīnmācīšanās principi | 33 ± 5 |
| Problēmas analīze, programmatūras specifikācija un darba plānošana, akcepttestēšana, atkļūdošana | 15 ± 5 |

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

## 2.3. Izziņas darbības līmenis

VPD iekļautie uzdevumi ir 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 VPD, apkopots 3. tabulā.

**3. tabula.** Izziņas darbības līmeņu raksturojums un to īpatsvars VPD.

| Izziņas darbības līmenis un tā apraksts | | Īpatsvars (%) |
|---|---|---|
| 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 VPD mērķi un galvenos vērtēšanas principus. 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. Otrais princips – SR veidu "Zināšanas un izpratne" un "Prasmju grupas" vērtēšanai iekļauti testelementi, kas atbilst III izziņas darbības līmenim, tādējādi akcentējot izpratnes vērtēšanu. Trešais princips – visi SR veida "Zināšanu, izpratnes, prasmju un ieradumu kombinācijas" vērtēšanai iekļautie uzdevumi ietver vismaz III līmenim atbilstošus testelementus.

# 3. Valsts pārbaudes darba uzbūve

VPD ir četras daļas. Daļu ilgums un starpbrīži:

- 1. daļa – 40 minūtes,
- starpbrīdis – 10 minūtes,
- 2. daļa – 40 minūtes,
- starpbrīdis – 40 minūtes,
- 3. daļa – 80 minūtes,
- starpbrīdis – 10 minūtes,
- 4. daļa – 80 minūtes.

Katrā eksāmena daļā iekļauti uzdevumi, kuri pārbauda satura moduļiem un sasniedzamo rezultātu veidiem atbilstošās SR grupas (sk. 4. tabulu).

1. un 2. daļā var tikt iekļauti dažāda veida uzdevumi, piemēram, atbilžu izvēles uzdevumi (viena pareizā atbilde), īso atbilžu uzdevumi, 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 VPD kopumā nav noteikts. Uzdevuma veida izvēli nosaka atbilstība sasniedzamajam rezultātam, ko tas pārbauda.

3. un 4. daļā ir iekļauti uzdevumi, kuros skolēnam jāizstrādā programmatūras produkts atbilstoši dotajam problēmas aprakstam, uzdevuma nosacījumiem. Skolēns drīkst izmantot jebkuru programmēšanas valodu un vidi, kuru ir apguvis mācību procesā skolā. Praktiskajās daļās tiks vērtēts programmēšanas labās prakses principu pielietojums (sk. 1. pielikumu). Ja VPD ir iekļauts temats par mašīnmācīšanās principiem, tas tiek pārbaudīts līdzīgi kā 1. un 2. daļā.

Pārbaudāmie sasniedzamie rezultāti katras daļas ietvaros gadu no gada var tikt mainīti.

**4. tabula.** VPD uzbūve.

| VPD daļa | | 1. daļa | 2. daļa | 3. daļa | 4. daļa | |
|---|---|---|---|---|---|---|
| SR veids | Satura modulis / SR grupa | Datortīkls un datubāze | Problēmas analīze, programmatūras specifikācija un darba plānošana, akcepttestēšana, atkļūdošana | Objektorientētā programmēšana un ārējās bibliotēkas | Datu struktūras, programmsaskarne (API) un mašīnmācīšanās principi | SR grupu īpatsvars (%) |
| **Zināšanas un izpratne** | Atpazīst pamatalgoritmus un izprot to darbības principus, spēj lasīt un izprast blokshēmās un pseidokodā rakstīto. Izprot kriptogrāfijas metožu nepieciešamību, datortīkla uzbūvi u. c. | | | | | 10 ± 5 |
| | Salīdzina programmatūras izstrādes modeļus, skaidro programmēšanas jēdzienus, raksturo mašīnmācīšanās izmantošanas iespējas u. c. | | | | | |
| **Prasmju grupas** | Izstrādā programmatūras prasību specifikāciju, programmatūru, izvēršanas plānu u. c., veic atkļūdošanu. | | | | | 57 ± 5 |
| | Lieto prasmes darbā ar informāciju. | | | | | 8 ± 3 |
| | Lieto programmēšanas labās prakses principus. | | | | | 3 ± 1 |
| **Zināšanu, izpratnes, prasmju un ieradumu kombinācijas** | Kombinē vairākus programmatūras dzīves cikla posmus: problēmas analīzi, programmatūras specifikāciju un darba plānošanu, programmatūras produkta izstrādi, akcepttestēšanu, atkļūdošanu, ieviešanu un uzturēšanu. Saskata algoritmu optimizācijas iespējas, izvērtē drošības riskus. | | | | | 16 ± 4 |
| | Satura moduļu un VPD daļu īpatsvars (%) | 15 ± 5 | 15 ± 5 | 33 ± 5 | 33 ± 5 | 100 |

# 4. Vērtēšanas kārtība un kritēriji

## 4.1. Vērtēšanas kārtība

Katrā uzdevumā ir norādīts maksimālais iegūstamo punktu skaits. Eksāmena vērtētājam ir pieejami kritēriji, pēc kuriem nosaka punktu skaitu, ko skolēns ieguvis. Skolēna rezultātus VPD – iegūto punktu summu visā darbā, iegūto punktu summu katrā daļā un iegūto punktu summu par noteiktu SR veidu vai grupu – izsaka procentuālā novērtējumā. Vidēji 20% VPD iekļauto testelementu reprezentē minimālo prasību kopumu – katra VPD satura moduļa izpildi atbilstoši 1. un 2. līmenim SOLO taksonomijā (piemēram, atpazīst pamatalgoritmus, nosauc piemērotāko izpētes metodi vai servera izveides pamatprincipus, apraksta objektorientētās programmēšanas darbības pamatprincipus. Izpilda vienkāršas darbības, piemēram, nodrošina datu izvadi. Kombinē zināšanas un prasmes standarta situācijās, piemēram, izveido masīvu datu uzglabāšanai).

## 4.2. Vērtēšanas kritēriji

Skolēnu sniegumu VPD 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).

# 5. Piekļuves nosacījumi

VPD var kārtot jebkurš skolēns, kas izstrādājis programmatūras produktu un tā dokumentāciju, īstenojot visus programmatūras izstrādes dzīves cikla posmus (līdz 15 A4 formāta lapām, neskaitot titullapu, satura rādītāju un pielikumu). Skolēns prezentē darbu un demonstrē izstrādāto programmatūras produktu. 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". Darbam jāsatur visas tālāk norādītās nodaļas. Katrā nodaļā dokumentācijas saturam jāatbilst vismaz SLA 2. līmenim "Turpina apgūt" (sk. 3. pielikumu) un programmatūras produktam jāatbilst tālāk norādītajām minimālajām prasībām. Dokumentācijas, programmatūras produkta un prezentācijas summatīvajam vērtējumam jābūt vismaz 4 (gandrīz viduvēji).

**Dokumentācija satur šādas nodaļas:**

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 programmatūras izstrādes plānošanu (bez darbietilpības novērtējuma). (T.A. 2.4.5.)

4. **Atkļūdošanas 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 vienkāršotu 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, kas veidots, ievērojot labās prakses principus (sk. 1. pielikumu). Programmatūras kods pievienojams dokumentācijas pielikumā.**
   SR: veido programmu, ievērojot labās prakses pieredzi tās pieraksta strukturēšanā un komentāru veidošanā. (T.A. 2.4.8.)

**Minimālās prasības programmatūras produktam:**

1. **Pielieto datu bāzi ar vairākām tabulām;**
   SR: 2.3.2. Plāno datubāzi, t. sk. izveido ER modeli konkrētā uzdevuma datu apstrādes risinājumam.
   SR: 2.4.17. Izveido vienkāršu datu apstrādes programmatūru (sistēmu), datu uzglabāšanai izmantojot paša veidotu datubāzi ar vairākām tabulām.

2. **Pielieto vismaz vienu bibliotēku un/vai API;**
   SR: 2.4.10. Izmanto programmēšanas valodas un tās bibliotēku dokumentāciju un palīdzības sistēmu, lai patstāvīgi apgūtu citas to piedāvātās iespējas, kas nepieciešamas konkrētās programmatūras izstrādei.
   SR: 2.4.11. Meklē un pievieno atvērtā koda bibliotēkas un lieto API (programmsaskarni) specializētu funkciju veikšanai sava programmēšanas projekta īstenošanai.
   SR: 2.4.13. Izvēlas un lieto atbilstošas programmēšanas valodas konstrukcijas, datu tipus un dažādas bibliotēkas, veidojot programmas doto uzdevumu un problēmu risinājumam.

3. **Pielieto vismaz vienu no minētajām datu struktūrām;**
   SR: 2.4.14. Izmanto dažādas datu struktūras (t. sk. masīvi, kopas, ieraksti, steks, rinda, saraksts, koks, grafs, datne) un ar tiem saistītos pamatalgoritmus.
   SR: 2.4.19. 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.

4. **Ir realizēta lietotāja piekļuves vai datu aizsardzība**.
   SR: 3.1.2. Izmanto kriptogrāfijas metodes konkrētā uzdevuma risinājumā.

# 6. Palīglīdzekļi, kurus atļauts izmantot valsts pārbaudes darba laikā

VPD 3. un 4. daļas programmēšanas uzdevumos:

1) Operētājsistēmas pamatprogrammas un biroja programmatūru (piemēram, kalkulatoru, failu pārlūku, izklājprogrammu), interneta pārlūkprogrammu (tikai atļautajām darbībām), izstrādes un atkļūdošanas vides un kompilatorus.
2) Valsts izglītības satura centram ir tiesības noteikt VPD norises laikā izmantojamās programmēšanas valodu mācību un programmatūru dokumentāciju vietnes, piemēram, "w3schools": W3Schools Online Web Tutorials vai citas.

# 7. Rīcības vārdu skaidrojums

VPD 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 VPD.

Rīcības vārdi ar piemēriem skaidroti 2. pielikumā.

**5. tabula.** Biežāk lietotie rīcības vārdi un to skaidrojums.

| Rīcības vārds | Skaidrojums |
|---|---|
| Analizē | Sīki, pamatīgi pēta, raksturo, piemēram, algoritmu, problēmas aprakstu. |
| Apvieno | Izveido vienu veselu (no atsevišķām vienībām, daļām, piemēram, apvieno programmatūras daļas). |
| Atpazīst | Pazīst, identificē (kādu/ko pēc noteiktām pazīmēm, parasti starp līdzīgiem, piemēram, OOP darbības principus). |
| Formulē | Izsaka (domu, atzinumu, lēmumu, jautājumu), piemēram, kādu programmatūras produktu nepieciešams izstrādāt. |
| Identificē | Nosaka, pazīst; konstatē (ko/kādu pēc kādām raksturīgām pazīmēm, piemēram, lomu sadalījumu projekta komandā). |
| Ievēro | Rīkojas saskaņā (ar ieteikumiem, norādījumiem u. tml.); iegaumē un izpilda (noteikumus, prasības u. tml.). Piemēram, labās prakses principus, rakstot programmatūras kodu. |
| Integrē | Iekļauj kādā vienībā; papildina, pilnveido, izveido par veselu vienību (piemēram, programmatūras daļas vienā veselumā). |
| Izmanto | Lieto (ko) par pamatu, materiālu (piemēram, kā noteikta gatavošanai, izveidošanai), piemēram, programmēšanas valodas dokumentāciju. |
| Izprot | Pilnīgi izzina un saprot, piemēram, algoritma darbību. |
| Izveido/Izstrādā | Ar mērķtiecīgu darbību panāk vai organizē, ka, piemēram, datubāze (vai kas cits) iegūst vēlamo veidu un atbilst noteiktām prasībām. Rada prasīto. |
| Izvēlas | Iegūst lietošanai, atrod, izraugās izmantošanai no kāda kopuma (ko piemērotu, atbilstošu, piemēram, programmatūras izstrādes modeli savam projektam atbilstoši problēmai, nosacījumiem u. tml.). |
| Izvērtē | Novērtē kādu kopumu, parasti pa tā sastāvdaļām, detaļām; katru faktu, notikumu un tml. noteiktā kopsakarībā (piemēram, mērķauditorijas specifiku un vajadzības). |
| Kombinē | Veido noteiktā veselumā (no atsevišķām sastāvdaļām), piemēram, vairākas algoritma daļas savienojot vienā veselumā. |
| Konfigurē | Pielāgo sistēmu (piemēram, datortīklu), uzstādot aparatūru un programmatūru (piemēram, maršrutētāja); noteiktā veidā uzstāda. |
| Lieto | Rīkojas (ar materiālu, koda fragmentu) mērķtiecīgi, pārdomāti – tā, lai gūtu labumu; izmanto (piemēram, lieto programmsaskarni). |
| Meklē | Cenšas (skatoties u. tml.) dabūt (ko vajadzīgu, nepieciešamu u. tml., piemēram, atvērtā koda bibliotēkas). |
| Pamato | Min faktus, cēloņus, loģiskus slēdzienus u. tml., kas pierāda (kā) patiesumu, nepieciešamību (piemēram, izvēli lietot konkrētu izpētes metodi lietotāju vajadzību noskaidrošanai). |
| Pārbauda | Pārliecinās, vai (kas) atrodas noteiktā stāvoklī, kārtībā (piemēram, programmatūras veiktspēju, programmatūras kodu), arī – vai kas ir noticis, izdarīts (piemēram, iegūts pareizais rezultāts, izpildot programmu). |
| Pielāgo | Veido, izveido (ko, piemēram, algoritmu) atbilstoši izmantošanas mērķim, apstākļiem. |
| Piemēro | Padara atbilstošu (piemēram, apstākļiem, iespējām programmatūras licenci). |
| Pievieno | Pieliek (pie kā, kam klāt, piemēram, atvērtā koda bibliotēkas). |
| Plāno | Veido (kāda objekta, piemēram, datubāzes) plānu, skici, shēmu. Iepriekš sadala (laiku) pa posmiem (kā veikšanai, piemēram, projekta īstenošanai). |
| Raksturo | Nosaka, apraksta, vērtē (kā) raksturīgās īpašības, pazīmes. Piemēram, mašīnmācīšanās algoritmus. |
| Salīdzina | Pēta, analizē, novēro u. tml. (divus vai vairākus priekšmetus, blokshēmas u. tml.), lai noteiktu (to) kopīgās vai atšķirīgās īpašības, pazīmes. |
| Saskata | Skatoties, pilnīgi, precīzi uztver (ko, piemēram, automatizācijas iespējas), atšķir, pazīst (ko); skatoties atklāj, konstatē (ko), secina (ko). |
| Sastāda | Izveido (piemēram, programmatūras prasību specifikāciju), apvienojot materiālus; uzraksta noteiktā formā, izmantojot kādus materiālus. |
| Skaidro | Stāstot, rakstot, arī rādot panāk, ka kādam (kas) kļūst zināms, saprotams (piemēram, programmatūras koda darbības principi). |
| Veic | Ar savu darbību, rīcību panāk, ka īstenojas (piemēram, kāda norise), tiek sasniegts (kāds mērķis). Piemēram, veic akcepttestēšanu. |

# PIELIKUMI

## 1. pielikums
## Programmēšanas labās prakses principu snieguma līmeņu apraksts

Labās prakses principi ir:

- katru atsevišķu priekšrakstu raksta jaunā rindā;
- koda loģiskās daļas (piemēram, zarošanos, ciklu, masīvu) savstarpēji atdala ar tukšu rindu;
- lieto atkāpes, lai vizualizētu priekšrakstu vai struktūru iekļaušanu citās struktūrās;
- izvairās no garām koda rindām. Lieto pārnesi jaunā rindā, atvieglojot koda lasīšanu;
- mainīgo, funkciju u. c. nosaukumus veido jēgpilnus, atvieglojot koda uztveri (piemēram, perimetru apzīmējot nevis ar "a", bet "perim" vai "perimetru");
- ar komentāriem skaidro programmatūras koda loģiskās daļas (piemēram, zarošanos, ciklu, masīvu), to lomu programmatūrā (piemēram "Datu izvade").

| Kritērijs/Līmenis | Sācis apgūt | Turpina apgūt | Apguvis |
|---|---|---|---|
| Veido programmatūru, ievērojot labās prakses principus tās pieraksta strukturēšanā un komentāru veidošanā. | 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. |

# 2. pielikums
# Rīcības vārdu skaidrojumi ar uzdevumu piemēriem

Tabulas virsrakstu skaidrojums:

- Sasniedzamais rezultāts (SR) – atbilstoši standartam.
- SOLO – maksimālais SR pārbaudes dziļums atbilstoši SOLO taksonomijai. Eksāmenā sasniedzamo rezultātu 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.

| Sasniedzamais rezultāts | SOLO | Rīcības vārdi | Uzdevumu veidu piemēri | Komentāri |
|---|---|---|---|---|
| 2.3.1. Izveido un konfigurē atvērtu vai aizsargātu daudzlietotāju lokālu tīklu, t. sk. veidojot atvērtus vai aizsargātus bezvadu piekļuves punktus. Izveido vienkāršu serveri un konfigurē piekļuvi tam no interneta. | 3 | Izveido tīklu | Uzzīmē un pamato datortīkla shēmu (piemēram, savienojot datorklases datorus, trīs skolotāju kabinetus, skolas WiFi, optiskā kabeļa ievadi), lietojot kabeļu, maršrutētāju, komutatoru, datoru, WiFi apzīmējumus. | Pārbaudāms teorētiski. |
| | | Konfigurē tīklu | Konfigurē maršrutētāju, iestatot drošas administratora un bezvadu tīkla paroles, piemērotu IP adrešu apgabalu. Konfigurē vecāku kontroles iestatījumus. | Pārbaudāms teorētiski. |
| | | Izveido un konfigurē serveri | Atpazīst, nosauc un skaidro servera izveides pamatprincipus (IP adresi, aparatūras un programmatūras konfigurāciju, drošības iestatījumus (karsto mijmaiņu (*hotswap*), ugunsmūri, telpas mikroklimatu, fizisko piekļuvi)). | Pārbaudāms teorētiski. |
| 2.3.2. Plāno datubāzi, t. sk. izveido ER modeli konkrētā uzdevuma datu apstrādes risinājumam. | 4 | Plāno | Uzzīmē uz papīra vai lietotnē (t. sk. izklājlapās) datubāzes shēmu ar vairākām tabulām un relācijām.<br>Piedāvā datubāzes pārvaldības sistēmu un pamato tās izvēli. | |
| | | Izveido | Izveido datubāzi.<br>Izveido datubāzes tabulas, konfigurē laukus.<br>Izveido relācijas.<br>Pārbauda, simulē datubāzes darbību, izmantojot SQL komandas *INSERT*, *SELECT*, *DELETE*, **JOIN*, *UPDATE*, *ORDER BY*. | |

| Sasniedzamais rezultāts | SOLO | Rīcības vārdi | Uzdevumu veidu piemēri | Komentāri |
|---|---|---|---|---|
| 2.4.1. Analizē dažādus ikdienas darba procesus, saskata tajos vai to daļās automatizācijas iespējas un kā pasūtītājs formulē darba uzdevumu programmatūras izstrādātājam. | 3 | Analizē un saskata | Iepazīstas ar problēmas aprakstu (teksta, audio, video formātā) un uzskaita dažādas procesu automatizācijas iespējas.<br>Ir dots problēmas apraksts un varianti automatizācijas risinājumiem – identificē, kuri no tiem atbilst dotajai situācijai un kuri neatbilst. | Dizaina domāšanas principi un soļi – izpēte, problēmas definēšana, ideju ģenerēšana. |
| | | Formulē | Balstoties uz dotu problēmas aprakstu, formulē, kādu programmatūras produktu nepieciešams izstrādāt.<br>Uzskaita un apraksta tehniskās prasības programmatūras produkta izstrādei. | |
| 2.4.2. Salīdzina un izvēlas piemērotāko programmatūras izstrādes modeli dotā uzdevuma atrisināšanai, pamato izvēli. | 3 | Salīdzina | Atpazīst un raksturo vienu vai vairākus programmatūras izstrādes modeļus.<br>Uzskaita ieguvumus, riskus konkrētu programmatūras izstrādes modeļu izmantošanā un salīdzina divus vai vairākus programmatūras izstrādes modeļus. | |
| | | Izvēlas un pamato | Ņemot vērā problēmas aprakstu, izvēlas un pamato, kuru programmatūras izstrādes modeli izmantot.<br>Plāno projekta izstrādes procesu, iekļaujot tajā programmatūras izstrādes modeli un paskaidrojumu par tā izvēli. | |
| 2.4.3. Izstrādā programmatūru grupā, ievērojot izvēlētā programmatūras izstrādes modeļa posmus, veic katra posma vienkāršotu dokumentēšanu. | 4 | Analizē, identificē, skaidro | Izlasa aprakstu, kurā atspoguļota programmatūras izstrādes projekta norise un dota informācija par iesaistītajiem cilvēkresursiem. Analizē doto aprakstu, identificē iesaistītās lomas, skaidro lomu nozīmi un pienākumus projektā.<br>Dots lomu saraksts. Skaidro, kurā no programmatūras izstrādes modeļa posmiem tiek iesaistītas konkrētas projektā iesaistītās lomas. | Iespēja dalīt atsevišķos uzdevumos, piemēram, tikai izskaidrot dotu projekta komandas lomu nozīmi un pienākumus programmatūras izstrādes projektā.<br>Kontekstā ar 2.4.5. – programmatūras izstrādes plānošana, darba pienākumu sadale grupā. |
| | | Plāno un pamato | Dots problēmas apraksts, programmatūras izstrādes projekta mērķis, projekta apjoms u. c. Skolēns plāno nepieciešamos cilvēkresursus – iesaistītās lomas, cilvēku skaitu u. c. Pamato izveidoto plānu. | Kontekstā ar 2.4.5. – programmatūras izstrādes plānošana, darba pienākumu sadale grupā. |
| | | Atpazīst un skaidro | Ir doti vairāku programmatūras izstrādes modeļa posmu procesa dokumentēšanas paraugi. Skolēns atpazīst, kuriem posmiem atbilst dokumentācijas paraugi, un skaidro, kādas pazīmes par to liecina. | |
| | | Izstrādā | Izstrādā dokumentāciju atsevišķiem programmatūras izstrādes modeļa posmiem, balstoties uz doto informāciju. | Pārbaudāms caur citiem saistītajiem SR: 2.4.4. (specifikācija), 2.4.5. (izstrādes plānošana), 2.4.6. (testēšanas plāns, testēšanas rezultāta dokumentācija), 2.4.7. (ieviešanas plāns, lietotāja ceļvedis). |

| Sasniedzamais rezultāts | SOLO | Rīcības vārdi | Uzdevumu veidu piemēri | Komentāri |
|---|---|---|---|---|
| 2.4.4. Sastāda vienkāršotu programmatūras prasību specifikāciju atbilstoši konkrētajam uzdevumam, izvērtējot mērķauditorijas specifiku un vajadzības. | 4 | Izvērtē | Definē un raksturo mērķauditoriju atbilstoši dotajam uzdevumam.<br>Atbilstoši dotajai problēmai identificē un apkopo informāciju par mērķauditorijas specifiku un vajadzībām.<br>Pielāgo programmatūras produkta ideju dažādām mērķauditorijām. | |
| | | Sastāda programmatūras prasību specifikāciju (PPS) | Balstoties uz dotu problēmas aprakstu, izstrādā pilnu PPS vai kādas tās daļas.<br>Papildina iesāktu PPS.<br>Analizē dotu PPS, iesaka savus papildinājumus, labojumus. | |
| 2.4.5. Veic vienkāršotu programmatūras projektēšanu (t. sk. lietotāju saskarnes un vienkāršotu datu modeļa izveidi) atbilstoši programmatūras prasību specifikācijā izvirzītajām funkcionālajām un nefunkcionālajām prasībām. Veic vienkāršotu programmatūras izstrādes plānošanu (bez darbietilpības novērtējuma) un darba pienākumu sadali grupā. | 3 | Plāno, skaidro, izstrādā | Pēc dotas programmatūras prasību specifikācijas veido programmatūras produkta lietotāja saskarnes papīra vai digitālu prototipu.<br>Shematiski attēlo programmatūras darbības principu, to skaidro. | Programmatūras izstrādes plānošana un darba pienākumu sadale atspoguļota pie 2.4.3. |
| 2.4.6. Veic programmatūras vienību izstrādi un vienībtestēšanu, izstrādājamās programmatūras vienību apvienošanu, integrācijas un akcepttestēšanu atbilstoši izstrādātajai programmatūras prasību specifikācijai un projektējuma aprakstam. | 4 | Izstrādā programmatūras vienības (daļas) | Izstrādā programmatūras vienības, piemēram, datubāzi, klienta daļu (*frontend*), servera daļu (*backend*). | |
| | | Pārbauda | Pārbauda programmatūras vienību veikstspēju (vienībtestēšana). Fiksē pārbaudes rezultātus. | |
| | | Apvieno, integrē | Apvieno, integrē programmatūras daļas, izveidojot vienotu programmatūras produktu. | |
| | | Veic akcepttestēšanu | Veic akcepttestēšanu atbilstoši izstrādātajām programmatūras prasībām.<br>Fiksē atbilstību. | |
| 2.4.7. Izstrādā un prezentē izveidotās programmatūras vienkāršotu izvēršanas (t. sk. ieviešanas) plānu, lietotāja ceļvedi un uzturēšanas plānu, ievērojot tās lietotāju mērķauditorijas specifiku. | 3 | Izstrādā | Izstrādā vienkāršotu programmatūras ieviešanas plānu (uzskaita, raksturo ieviešanas soļus, nepieciešamos resursus). Pamato plāna atbilstību mērķauditorijai.<br><br>Pielāgo esošu ieviešanas plānu dotajai mērķauditorijai. Pamato veiktās izmaiņas. | |

| Sasniedzamais rezultāts | SOLO | Rīcības vārdi | Uzdevumu veidu piemēri | Komentāri |
|---|---|---|---|---|
| 2.4.8. Veido programmu, ievērojot labās prakses pieredzi tās pieraksta strukturēšanā un komentāru veidošanā, vienojoties par vienotu stilu ar visiem grupas dalībniekiem. | 4 | Ievēro labās prakses principus | Veido programmatūru, ievērojot labās prakses principus:<br>• katru atsevišķu priekšrakstu raksta jaunā rindā;<br>• koda loģiskās daļas (piemēram, zarošanos, ciklu, masīvu) savstarpēji atdala ar tukšu rindu;<br>• lieto atkāpes, lai vizualizētu priekšrakstu vai struktūru iekļaušanu citās struktūrās;<br>• izvairās no garām koda rindām. Lieto pārnesi jaunā rindā, atvieglojot koda lasīšanu;<br>• mainīgo, funkciju u.c. nosaukumus veido jēgpilnus, atvieglojot koda uztveri (piemēram, perimetru apzīmējot nevis ar "a", bet "perim" vai "perimetru");<br>• ar komentāriem skaidro programmatūras koda loģiskās daļas (piemēram, zarošanos, ciklu, masīvu), to lomu programmatūrā (piemēram "Datu izvade"). | |
| 2.4.9. Lieto projektu un versiju pārvaldības rīkus sadarbībai ar citiem programmatūras izstrādes procesā un tā vadībā. | 3 | Skaidro, pamato | Skaidro, kas ir versiju pārvaldības rīks un kāpēc to izmanto.<br>Skaidro un pamato versiju pārvaldības rīku nepieciešamību programmatūras izstrādes procesā un tā vadībā. Nosauc ieguvumus, riskus, piemērus. | |
| 2.4.10. Izmanto programmēšanas valodas un tās bibliotēku dokumentāciju un palīdzības sistēmu, lai patstāvīgi apgūtu citas to piedāvātās iespējas, kas nepieciešamas konkrētās programmatūras izstrādei. | 4 | Analizē un izmanto citas piedāvātās iespējas | Izstrādājot programmatūru, jēgpilni izmanto esošo bibliotēku piedāvātās iespējas.<br>Lieto esošo klašu konstruktorus un to metodes, tādējādi saīsinot esošo programmatūras kodu un uzlabojot izstrādājamās programmatūras darba efektivitāti.<br>Analizē doto bibliotēku dokumentāciju un palīdzības sistēmu informāciju, lai atrastu un pielietotu atbilstošu bibliotēku problēmas risināšanā.<br>Izmanto programmēšanas valodas dokumentāciju, lai apgūtu programmēšanas valodas iespējas dotās problēmas risināšanā (t. sk. sintakses pārbaudē). | Piemērs: dots neapgūtas programmēšanas valodas koda fragments. Jānovērš kļūda, izmantojot programmēšanas valodas dokumentāciju (piemēram, PHP zarošanos). |
| 2.4.11. Meklē un pievieno atvērtā koda bibliotēkas un lieto API (programmsaskarni) specializētu funkciju veikšanai sava programmēšanas projekta īstenošanai. | 4 | Meklē | Atrod atvērtā koda bibliotēku, kas veic nepieciešamo funkciju, un pamato izvēli. | Piemēram, pamato atbildi ar saiti un citātu no dokumentācijas. |
| | | Pievieno | Pievieno paša atrastu bibliotēku.<br>Pievieno dotu bibliotēku. | |
| | | Lieto | Pielieto bibliotēku un/vai API dotā uzdevuma risinājumam. | Piemēram, sasaista ar datubāzi. |

| Sasniedzamais rezultāts | SOLO | Rīcības vārdi | Uzdevumu veidu piemēri | Komentāri |
|---|---|---|---|---|
| 2.4.12. Izvēlas programmēšanas valodu un programmatūras izstrādes vidi programmatūras izstrādē atbilstoši uzdevuma specifikai, pamato savu izvēli. | 3 | Izvēlas un pamato | Izvēlas konkrētā uzdevuma specifikai atbilstošāko programmatūras izstrādes valodu.<br>Pamato programmēšanas valodas izvēli, to salīdzinot ar citas programmēšanas valodas piemērotību konkrētā uzdevuma atrisināšanai.<br>Salīdzina divas programmatūras izstrādes vides, nosaucot priekšrocības un trūkumus. | |
| 2.4.13. Izvēlas un lieto atbilstošas programmēšanas valodas konstrukcijas, datu tipus un dažādas bibliotēkas, veidojot programmas doto uzdevumu un problēmu risinājumam. | 4 | Izvēlas | Izvēlas atbilstošas programmēšanas valodas konstrukcijas, datu tipus, bibliotēkas dotā uzdevuma risinājumam. | Piemēram, uzzīmē blokshēmu, uzraksta pseidokodu, sakārto dotos koda moduļus pareizā secībā. |
| | | Lieto, veidojot programmu | Izstrādā risinājumu dotajam uzdevumam, iekļaujot atbilstošas programmēšanas valodas konstrukcijas, datu tipus, bibliotēkas. | |
| 2.4.14. Izmanto dažādas datu struktūras (t. sk. masīvi, kopas, ieraksti, steks, rinda, saraksts, koks, grafs, datne) un ar tiem saistītos pamatalgoritmus. | 3 | Izmanto | Izveido risinājumu dotajam uzdevumam, izmantojot viendimensijas un divdimensiju masīvus, datnes. | Citu datu struktūru pielietojumu apskata mācību procesā. |
| 2.4.15. Skaidro objektorientētās programmēšanas pamatprincipus, veido programmas vienā no objektorientētajām programmēšanas valodām. | 4 | Skaidro OOP pamatprincipus | Skaidro klašu, īpašību un metožu izveides un pielietošanas iespējas:<br>• uz papīra vai lietotnē uzraksta pseidokodu ar piemēriem (dota problēma);<br>• atpazīst un skaidro pseidokoda piemēra darbības principus (dots pseidokods);<br>• pamato pseidokoda optimizāciju, ja lietotu OOP pamatprincipus (dots pseidokods). | |
| | | Veido programmas | Veido programmatūras produktu, balstoties uz OOP pamatprincipiem – izveido un pielieto klasi un tās īpašības, metodes un konstruktorus. | |
| 2.4.16. Veic izstrādātajā programmatūrā izmantoto algoritmu sarežģītības novērtēšanu pēc laika un izmantotās atmiņas un, ja nepieciešams, veic algoritmu optimizāciju. | 4 | Veic novērtēšanu | Novērtē algoritma izpildes laiku dažādās situācijās, salīdzina vismaz divu dažādu algoritmu izpildes laiku (darbību skaitu). | Piemēram, ir doti divi kārtošanas algoritmi, to apraksts un vizualizācija (burbuļa, ievietošanas vai tml.). |
| | | Veic optimizāciju | Optimizē datu tipus. Pielieto lokālos un globālos mainīgos.<br>Samazina programmatūras izpildes laiku, optimizējot/aizvietojot koda daļas (piemēram, „if” ar „switch”, likvidē atkārtotas to pašu datu pārbaudes). | |

| Sasniedzamais rezultāts | SOLO | Rīcības vārdi | Uzdevumu veidu piemēri | Komentāri |
|---|---|---|---|---|
| 2.4.17. Izveido vienkāršu datu apstrādes programmatūru (sistēmu), datu uzglabāšanai izmantojot paša veidotu datubāzi ar vairākām tabulām. | 4 | Izveido | Izveido programmatūru, lai pieslēgtos dotai datubāzes datnei. Apstrādā datubāzes datus – pievieno, dzēš, rediģē, ielasa. | Apgūst kontekstā ar 2.3.2<br>Piemēram, dots MDF fails, C# vai Pyodbc u. tml. |
| 2.4.18. Salīdzina mašīnmācīšanās algoritmus (t. sk. attēlu atpazīšanai) un raksturo to izmantošanas iespējas jaunas programmatūras izstrādē. Izmanto bibliotēku, kurā realizēti mašīnmācīšanās algoritmi, izstrādājot programmatūru mācību uzdevuma realizācijai. | 3 | Salīdzina | Ir dots problēmas apraksts un divi mašīnmācīšanās algoritmi tās risināšanai. Nosaka optimālāko algoritmu un pamato izvēli. | |
| | | Raksturo | Ir dots konkrētas jomas vai situācijas mašīnmācīšanās algoritma apraksts. Raksturo ieguvumus un riskus (piemēram, kļūdu vai paroles atpazīšana teksta ievades brīdī). | |
| | | Izmanto, izstrādājot programmatūru | | Eksāmenā netiek pārbaudīts! |
| 2.4.19. 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. | 4 | Izmanto gatavus | Izvēlas no dotajiem algoritmiem piemērotāko dotās problēmas risinājumam. | |
| | | Pielāgo | Pielāgo (maina) doto algoritmu, lai tas risinātu doto problēmu. | |
| | | Kombinē | Kombinē divus vai vairākus algoritmus (vai to daļas), lai izveidotu risinājumu dotajai problēmai. | |
| | | Izstrādā jaunus | Veido risinājumu visai problēmai vai tās daļai (piemēram, tikai datu izvadei pseidokodā vai konkrētā valodā). | Programmēšanas prasmes pārbauda arī citu SR kontekstā. |
| | | Izprot un skaidro algoritmu | Skaidro dotā algoritma blokshēmu, pseidokodu vai programmatūras kodu. | |
| | | Izprot un pielāgo | Papildina dotā algoritma blokshēmu, pseidokodu vai programmatūras kodu, lai tas risinātu doto problēmu. Ja nepieciešams, veido jaunas datu struktūras (piemēram, masīvu, pievieno datni). | |
| 3.1.2. Izmanto kriptogrāfijas metodes konkrētā uzdevuma risinājumā. | 3 | Izmanto kriptogrāfijas metodes | Pielieto kriptogrāfijas metodes (piemēram, ievietojot ierakstu datubāzē ar šifrētu paroli).<br>Skaidro jēdzienus “privātā” un “publiskā” atslēga. Nosauc to pielietojumu. | Izprot kriptogrāfijas metožu nepieciešamību (paroles, sensitīvie dati, *frontend-backend* komunikācija (JWT (*JSON Web Token*)). Dubultā autentifikācija. |
| 3.1.4. Salīdzina atvērto kodu licences un to atšķirības, izmanto un piemēro atbilstošāko no licencēm savam programmatūras projektam. | 3 | Salīdzina un piemēro atbilstošāko no licencēm | Doti licenču piemēri. Izvēlas dotajām programmatūras prasībām atbilstošāko licenci. Pamato izvēli. | |

# 3. pielikums
# Projekta izstrādes dokumentācijas snieguma līmeņu apraksts

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

| Kritērijs/ Līmenis | Sācis apgūt | Turpina apgūt | Apguvis | Apguvis padziļināti |
|---|---|---|---|---|
| Problēmas izpēte un analīze | | | | |
| Izpētes metodes | Nosauc un raksturo izpētes metodes (piemēram, interviju, priekšizpēti, novērojumu u. c.). Trūkst izpētes metožu pozitīvo iezīmju un trūkumu uzskaitījums. Izpētes metodes pamatojumam nav saistība ar doto problēmu. | Nosauc un raksturo izpētes metodes, minot katras izpētes metodes pozitīvās iezīmes un trūkumus. Izpētes metodes izvēles pamatojums daļēji ir saistīts ar doto problēmu. | Izpētes metodes izvēles pamatojums ir balstīts uz pozitīvo iezīmju un trūkumu salīdzinājumu un ņemot vērā doto problēmu. | |
| Izpētes process | 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 (vai tml.). | Aprakstā iekļautā izpētes procesa gaita ir loģiska, detalizēta un sastāv no vismaz pieciem soļiem – piemēram, sagatavošanās, norise, rezultātu apkopošana, izvērtēšana un secinājumu izdarīšana. | |
| Izpētes dati | Pievienotie dati daļēji atbilst pētāmajai problēmai un ir pievienoti pielikumā. | Dati atbilst pētāmajai problēmai un ir pievienoti pielikumā. | Dati atbilst pētāmajai problēmai, ir pievienoti pielikumā un ir strukturēti. | |
| Programmatūras prasību specifikācija | | | | |
| Programmatūras produkta mērķauditorija | Mērķauditorija ir definēta, neiedziļinoties dotajā problēmā. | Ir definēta, tomēr iespējams to konkretizēt. | Tiek definēta konkrēta risinājuma mērķauditorija. | |
| Mērķauditorijas vajadzību raksturojums | Vispārīgs, neiedziļinoties mērķauditorijas vajadzībās. | Daļa no vajadzībām ir saistītas ar mērķauditoriju un doto problēmu. | Vajadzības ir saistītas ar mērķauditoriju un doto problēmu. | |
| Programmatūras produkta apraksts | Ļoti vispārīgs. Nesniedz ieskatu, kādas tehnoloģijas būtu jāizmanto. | Ir aprakstīts, kas ir izstrādājamais programmatūras produkts, tomēr nav konkrēti uzskaitītas tehnoloģijas, kas tiks izmantotas tā izstrādē. | Ir aprakstīts, kas ir izstrādājamais programmatūras produkts un kādas tehnoloģijas nepieciešamas tā izveidē. | |

| | | | | |
|---|---|---|---|---|
| Programmatūras produkta funkciju apraksts | Funkciju apraksts ir vispārīgs. | Funkcijas ir nosauktas (piemēram, lietotāja autentifikācija, datu atlases filtrs u. tml.), raksturotas un daļēji atbilst mērķauditorijas vajadzībām. | Funkcijas ir nosauktas, raksturotas un atbilst mērķauditorijas vajadzībām. | |
| Programmatūras produkta skice | Programmatūras produkta skice daļēji atbilst aprakstam un funkciju prasībām. Tajā iekļautā informācija ir grūti uztverama. | Programmatūras produkta skice daļēji atbilst aprakstam un funkciju prasībām. | Atbilst aprakstam un funkciju prasībām. Atbilstoši konkrētā darba vajadzībām izstrādāts: programmatūras struktūras, procesu plūsmas skice; lietotāja saskarnes skice; datubāzes skice. | Papildu izstrādāts: datu struktūras skice; datu uzglabāšanas un drošības skice. |
| Programmatūras izstrādes plāns | | | | |
| Programmatūras izstrādes plāns | Izstrādāts plāns, norādot programmatūras izstrādes dzīves cikla posmus. Atsevišķiem programmatūras dzīves cikla posmiem norādīti galvenie veicamie uzdevumi vai to nav vispār. | Izstrādāts plāns, norādot atsevišķiem programmatūras izstrādes dzīves cikla posmiem paredzēto laiku. Atsevišķiem programmatūras dzīves cikla posmiem norādīti galvenie veicamie uzdevumi vai to nav vispār. | Izstrādāts plāns, norādot katram programmatūras izstrādes dzīves cikla posmam paredzēto laiku. Katram programmatūras dzīves cikla posmam norādīti galvenie veicamie uzdevumi. Plāns ir realizējams paredzētajā laikā. | Izstrādāts plāns, norādot katram programmatūras izstrādes dzīves cikla posmam paredzēto laiku. Katram programmatūras dzīves cikla posmam norādīti galvenie veicamie uzdevumi, apakšuzdevumi un tiem paredzētais laiks. |
| Atkļūdošanas un akcepttestēšanas pārskats | | | | |
| Testēšanas plāns un izpilde | Testēšanas plāns paredz pārbaudīt tikai atsevišķas sistēmas daļas. | Testēšanas plāns ir nepilnīgs, tajā trūkst informācijas par atsevišķu sistēmas daļu testēšanu. Ne visas specifikācijā minētās programmatūras produkta funkcijas tiek pārbaudītas. | Testēšanas plānā noteikts, kuras sistēmas daļas un kādā veidā jātestē. Nosaka nepieciešamo rezultātu atbilstoši specifikācijā esošajam programmatūras produkta funkciju aprakstam. | Testēšanas plānā noteikts, kuras sistēmas daļas un kādā veidā jātestē. Nosaka nepieciešamo rezultātu atbilstoši programmatūras produkta funkciju aprakstam. Satur plāna izpildes atspoguļojumu (piemēram, ekrānuzņēmumus no testēšanas procesa). |
| Akcepttestēšanas pārskats | Akcepttestēšanas pārskats ir vispārīgs un satur virspusēju programmatūras prasību specifikācijā esošo prasību pārbaudi no lietotāja skatu punkta. | Akcepttestēšanas pārskats ir vispārīgs un satur lielāko daļu programmatūras prasību specifikācijā esošo prasību pārbaudi no lietotāja skatu punkta. | Atspoguļota sistēmas darbības pārbaude no lietotāja skatu punkta atbilstoši specifikācijai. | Atspoguļota sistēmas darbības pārbaude no lietotāja skatu punkta atbilstoši specifikācijai. Atspoguļota sistēmas uzlabojumu veikšana un/vai papildu funkciju izstrāde atbilstoši lietotāju vēlmēm. |

| | | | | |
|---|---|---|---|---|
| Lietotāja ceļvedis | | | | |
| Lietotāja ceļvedis | Izveidots lietotāja ceļvedis, kura struktūra ir juceklīga. Dokumenta aturs vispārīgi apskata daļu programmatūras produkta funkciju. | Izveidots lietotāja ceļvedis:<br>kam daļēji pievienota vizuāla informācija;<br>dokumenta struktūra un saturs ir juceklīgs. | Izveidots lietotāja ceļvedis, kas atbilst šādiem kritērijiem:<br>pievienota vizuāla informācija;<br>loģiska dokumenta struktūra, viegli saprotamas sadaļas;<br>iespēja viegli atrast nepieciešamo informāciju. | Izveidots lietotāja ceļvedis, kas atbilst šādiem kritērijiem:<br>iespējami vienkārša un saprotama valoda;<br>pievienota vizuāla informācija;<br>loģiska dokumenta struktūra, viegli saprotamas sadaļas;<br>iespēja viegli atrast nepieciešamo informāciju;<br>iekļauta tikai nepieciešamā informācija. |
| Piemērotās licences pamatojums | | | | |
| Produkta licence | Licence ir norādīta, bet tās izvēle nav pamatota. | Norādīta licence, tās izvēles pamatojums ir vispārīgs. | Norādīta atbilstoša licence un tās izvēle ir pamatota (piemēram, specifikācijā). | |
| Programmatūras kods, kas veidots ievērojot labās prakses principus (1. pielikumā) | | | | |
| Veido programmatūru, ievērojot labās prakses principus tās pieraksta strukturēšanā un komentāru veidošanā | Lielākajā daļā programmatūras koda lieto labās prakses principus, bet nekonsekventi vai daļēji korekti. | Kopumā korekti un konsekventi programmatūras kodā lieto labās prakses principus, pieļaujot dažas neprecizitātes. | Programmatūras kodā labās prakses principus lieto korekti un konsekventi. | |

DOMĀT.
DARĪT.
ZINĀT.

**Valsts izglītības satura centra īstenotā projekta "Kompetenču pieeja mācību saturā" mērķis ir izstrādāt, aprobēt un pēctecīgi ieviest Latvijā tādu vispārējās izglītības saturu un pieeju mācīšanai, lai skolēni gūtu dzīvei 21. gadsimtā nepieciešamās zināšanas, prasmes un attieksmes.**

Projekts Nr. 8.3.1.1/16/I/002 Kompetenču pieeja mācību saturā

Valsts izglītības satura centrs | NACIONĀLAIS ATTĪSTĪBAS PLĀNS 2020 | EIROPAS SAVIENĪBA Eiropas Sociālais fonds

IEGULDĪJUMS TAVĀ NĀKOTNĒ