2.9 Programmēšana: atbalsts izvēlētos uzdevumos

Mācīties programmēt, neizmantojot MI rakstīto kodu

Priekšmets: Programmēšana · **Klase:** 9. (14–15 gadi) · **Ilgums:** 1 dubultstunda (90 min) · **MI:** Atbalsts **AR** MI

Ar MI saistītās kompetences (DUAL.AI.TEACHer ietvars) – skolotāja līmenis	Priekšmetam specifiski mācīšanās mērķi – skolēna/apguvēja līmenis
AF-TL-2b *(MI pamati un pielietojumi × Mācīšana un mācīšanās, 2. līmenis – Refleksīva ieviešana)*: "Skolotāji var organizēt nodarbības elementus, kas ietver MI izmantošanu ar skaidrām pedagoģiskām lomām un stratēģijām rezultāta elementu pārbaudei." **AF-FC-2b** *(MI pamati un pielietojumi × Apguvēju (MI) digitālās kompetences veicināšana, 2. līmenis – Refleksīva ieviešana)*: "Skolotāji var īstenot norādes, kas palīdz apguvējiem attīstīt praktiskas MI izmantošanas prasmes, pielāgojot uzdevumus apguvēju iepriekšējām zināšanām un pieejamajiem MI rīkiem."	Stundas beigās skolēni spēj: • izmantot MI atkārtotiem vai atbalsta uzdevumiem programmēšanā, piemēram, koda atkāpju sakārtošanai vai komentāru rakstīšanai; • lūgt MI modelim ieteikt koda uzlabojumus, nesniedzot pašu kodu, lai skolēni paši varētu izvērtēt savas zināšanas; • kritiski izvērtēt MI modeli, jautājot tam, kā tas optimizētu doto kodu.

> **Galvenā atziņa:** Skolēniem ir jāzina programmēšanas pamati, ja viņi vēlas izvērtēt modeļa sniegtos rezultātus. Tāpēc skolēniem, kuri sper pirmos soļus programmēšanas apguvē, MI modeļi jāizmanto ar programmēšanu saistītiem atbalsta un atkārtotiem uzdevumiem, bet ne koda radīšanai. Viņi var arī

> izmantot modeli, lai jautātu par iespējamiem koda uzlabojumiem, un tādā veidā pārbaudīt savas programmēšanas prasmes.



1. att.: "Mācīties programmēt, neizmantojot MI rakstīto kodu MI rakstīto kodu" (MI ģenerēts attēls, Google Gemini 3.6 Thinking)

Saturs

Programmēšanas izglītība mainās apstākļos, kuros MI sistēmas spēj ģenerēt, skaidrot un pārskatīt kodu. Lai gan šie rīki piedāvā jaunas mācīšanās iespējas, tie rada arī būtisku izglītības izaicinājumu: skolēniem ir nepieciešamas pietiekamas programmēšanas zināšanas, lai izvērtētu, vai MI ģenerētais ieteikums ir pareizs, efektīvs vai vispār atbilstošs konkrētajam uzdevumam. Bez programmēšanas jēdzienu pamatizpratnes skolēni var pieņemt nepareizus risinājumus, nepamanīt kļūdas vai neatpazīt labākas alternatīvas. Tāpēc programmēšanas pamatu apguve joprojām ir būtiska arī tad, ja ir pieejami MI rīki.

Programmēšanas ievadkursos MI būtu jāpozicionē kā mācīšanās atbalsta rīks, nevis kā programmētāja aizstājējs. Skolēni var efektīvi izmantot MI atkārtotiem vai zema līmeņa uzdevumiem, kas nesamazina programmēšanas apguvē ieguldāmo kognitīvo piepūli. Piemēri ir koda formatēšana un atkāpju sakārtošana, dokumentācijas komentāru ģenerēšana, kļūdu paziņojumu skaidrošana, sintakses noteikumu apkopošana vai palīdzība skolēniem izprast konkrēto programmēšanas konstrukciju nozīmi. Šādi izmantojumi ļauj skolēniem koncentrēt uzmanību uz pamatā esošo loģiku, algoritmiem un problēmu risināšanas procesiem, kas ir programmēšanas kompetences kodols.

Vēl viens produktīvs MI izmantošanas veids ir koda pārskatīšana. Tā vietā, lai lūgtu MI modelim uzrakstīt risinājumu, skolēni vispirms var uzrakstīt paši savu kodu un pēc tam lūgt modelim norādīt iespējamos uzlabojumus. MI var ieteikt skaidrākus mainīgo nosaukumus, labāku koda organizāciju, mazināt liekvārdību, uzlabot lasāmību vai sniegt alternatīvas pieejas, ko skolēns var patstāvīgi izvērtēt. Tādējādi MI kļūst par atgriezeniskās saites partneri, ļaujot skolēniem salīdzināt savu spriešanu ar ārējiem ieteikumiem un novērtēt savu risinājumu stiprās un vājās puses.

Tomēr MI ģenerētu atgriezenisko saiti nevajadzētu pieņemt bez kritikas. Modeļi var ieteikt izmaiņas, kas nav vajadzīgas, ir neefektīvas vai balstītas uz nepareiziem pieņēmumiem par koda mērķi. Tāpēc šīs stundas galvenais mērķis ir attīstīt skolēnu spēju kritiski izvērtēt MI ieteikumus. Skolēni mācīsies jautāt MI modelim ne tikai to, kādus uzlabojumus tas ierosina, bet arī to, kāpēc tas uzskata šīs izmaiņas par lietderīgām. Salīdzinot modeļa spriešanu ar vispāratzītiem programmēšanas principiem, skolēni var spriest, vai ierosinātā optimizācija patiešām uzlabo kodu vai arī tā ir tikai cita dizaina izvēle.

Šajā stundā MI tiek uzskatīts par rīku, kas atbalsta refleksiju, pašnovērtējumu un koda kvalitātes uzlabošanu, nevis koda ģenerēšanu. Skolēni vingrinās rakstīt paši savas programmas, izmantot MI atbalsta uzdevumiem, lūgt atgriezenisko saiti par savu darbu un kritiski analizēt saņemto atbilžu kvalitāti. Galvenais mērķis ir palīdzēt skolēniem kļūt gan par labākiem programmētājiem, gan par informētākiem MI sistēmu lietotājiem.

Stundas plāns

Fāze	Laiks	Darbība
Ievads	10 min	Skolotājs parāda divus īsus koda risinājumus vienai un tai pašai problēmai – vienu, ko uzrakstījis skolēns, un otru, ko ģenerējis MI modelis. Skolēni pārrunā, kurš risinājums šķiet skaidrāks, efektīvāks vai vieglāk saprotams.
Ievade	10 min	Skolotājs iepazīstina ar piemērotiem MI izmantošanas veidiem programmēšanas izglītībā, uzsverot atšķirību starp atbalsta uzdevumiem, koda pārskatīšanu un koda ģenerēšanu.
Izpēte	15 min	Skolēni pāros strādā pie neliela programmēšanas uzdevuma un nosaka, kurus uzdevumus varētu pamatoti uzticēt MI asistentam un kuriem jāpaliek viņu pašu atbildībā.
Uzlabošanas cikls	35 min	Skolēni uzraksta paši savu risinājumu un izpilda 2–3 kārtas pēc principa rakstīt → jautāt → izvērtēt → pārstrādāt, izmantojot MI tikai atbalsta uzdevumiem un uzlabojumu ieteikumiem.
Refleksija	20 min	Diskusija ar visu klasi par MI atgriezeniskās saites lietderību un ierobežojumiem, pievēršot uzmanību tam, kā skolēni noteica, vai ieteikumi bija vērtīgi vai nē.

Vadošie jautājumi: sarežģītības slāņošana

35 minūšu uzlabošanas cikla laikā skolēni izpilda tik daudz kārtu, cik atļauj laiks. Katrā kārtā tiek ievērots viens un tas pats četru soļu process: rakstīt → jautāt → izvērtēt → pārstrādāt

1. **1. kārta — MI izmantošana atbalsta uzdevumiem**
   - Rakstīt: Patstāvīgi izpildiet īsu programmēšanas uzdevumu.
   - Jautāt: Lūdziet MI atbalstu ar kodu saistītam uzdevumam, kas nav programmēšana, piemēram:
     - atkāpju uzlabošanai,
     - komentāru ģenerēšanai,
     - kļūdas paziņojuma skaidrošanai,
     - funkcijas darbības aprakstīšanai.
   - Izvērtēt: Vai MI skaidrojums precīzi apraksta kodu? Vai ģenerētie komentāri ir noderīgi un saprotami?
   - Pārstrādāt: Iekļaujiet noderīgās izmaiņas, nodrošinot, ka programmas loģika pilnībā paliek jūsu pašu darbs.
2. **2. kārta — Uzlabojumu ieteikumu pieprasīšana**
   - Rakstīt: Pārskatiet savu pabeigto programmu un nosakiet sadaļas, kuras varētu uzlabot.
   - Jautāt: Lūdziet uzlabojumu ieteikumus, neļaujot MI pārrakstīt kodu. Piemēram:

"Iesaki trīs veidus, kā uzlabot šīs programmas lasāmību un uzturamību, bet nesniedz aizstājošu kodu."

   - Izvērtēt: Kuri ieteikumi šķiet vērtīgi? Kuriem ieteikumiem būtu maza ietekme uz programmu?
   - Pārstrādāt: Ieviesiet uzlabojumus, kuriem piekrītat, un pārbaudiet, vai programma joprojām darbojas pareizi.
2. **3. kārta — Optimizācijas ieteikumu kritiska izvērtēšana**
   - Rakstīt: Izvēlieties sava koda daļu, kas veic atkārtotu uzdevumu vai izmanto ciklus.
   - Jautāt: Lūdziet optimizācijas ieteikumus un pieprasiet, lai MI pamato savus ieteikumus.

Uzvednes piemērs:

"Paskaidro, kā šo kodu varētu optimizēt un kāpēc katra ierosinātā optimizācija uzlabotu veiktspēju vai lasāmību. Nesniedz optimizēto kodu."

     - Izvērtēt: Vai ierosinātās optimizācijas patiešām ir lietderīgas? Kuras no tām ir pamatotas ar skaidru argumentāciju? Kuras šķiet nevajadzīgas vai grūti pamatojamas?
     - Pārstrādāt: Izlemiet, kurus ieteikumus ieviest, un paskaidrojiet savu pamatojumu.

Refleksijas jautājumi:

- Kuri MI ieteikumi bija visnoderīgākie jūsu koda uzlabošanai?
- Vai MI sniedza kādus ieteikumus, kuriem jūs nepiekritāt? Kāpēc?
- Kā jūs noteicāt, vai optimizācija ir tā vērta?

- Kad MI ir visnoderīgākais programmēšanas procesā?
- Kuri programmēšanas uzdevumi joprojām būtu jāveic skolēnam?
- Kāpēc programmēšanas zināšanas ir nepieciešamas, lai izvērtētu MI ģenerētu atgriezenisko saiti?
- Kā skolēni var izvairīties no pārmērīgas atkarības no MI rīkiem, mācoties programmēt?

Noslēguma jautājums (refleksijas fāzei):

Cik daudz programmēšanas zināšanu ir nepieciešams, lai noteiktu, vai MI ieteikums patiešām uzlabo programmu, un kādi riski rodas, ja lietotāji paļaujas uz MI ģenerētiem ieteikumiem, paši nesaprotot kodu?

Materiāli

- Viens klēpjdators katram skolēnu pārim ar programmēšanas vidi
- Piekļuve MI tērzēšanas asistentam vai lokāli instalētam valodas modelim
- Īss programmēšanas uzdevums, kas atbilst skolēnu pieredzes līmenim
- Darba lapa, kas seko ciklam rakstīt → jautāt → izvērtēt → pārstrādāt
- Refleksijas jautājumi, kas vērsti uz MI pratību un kritisku izvērtēšanu
- Pēc izvēles – koda piemēri, kas ilustrē labu un sliktu programmēšanas praksi

---

Revision #4
Created 2026-10-20 15:29:53 UTC by Alexander
Updated 2026-10-29 14:16:23 UTC by Jannik