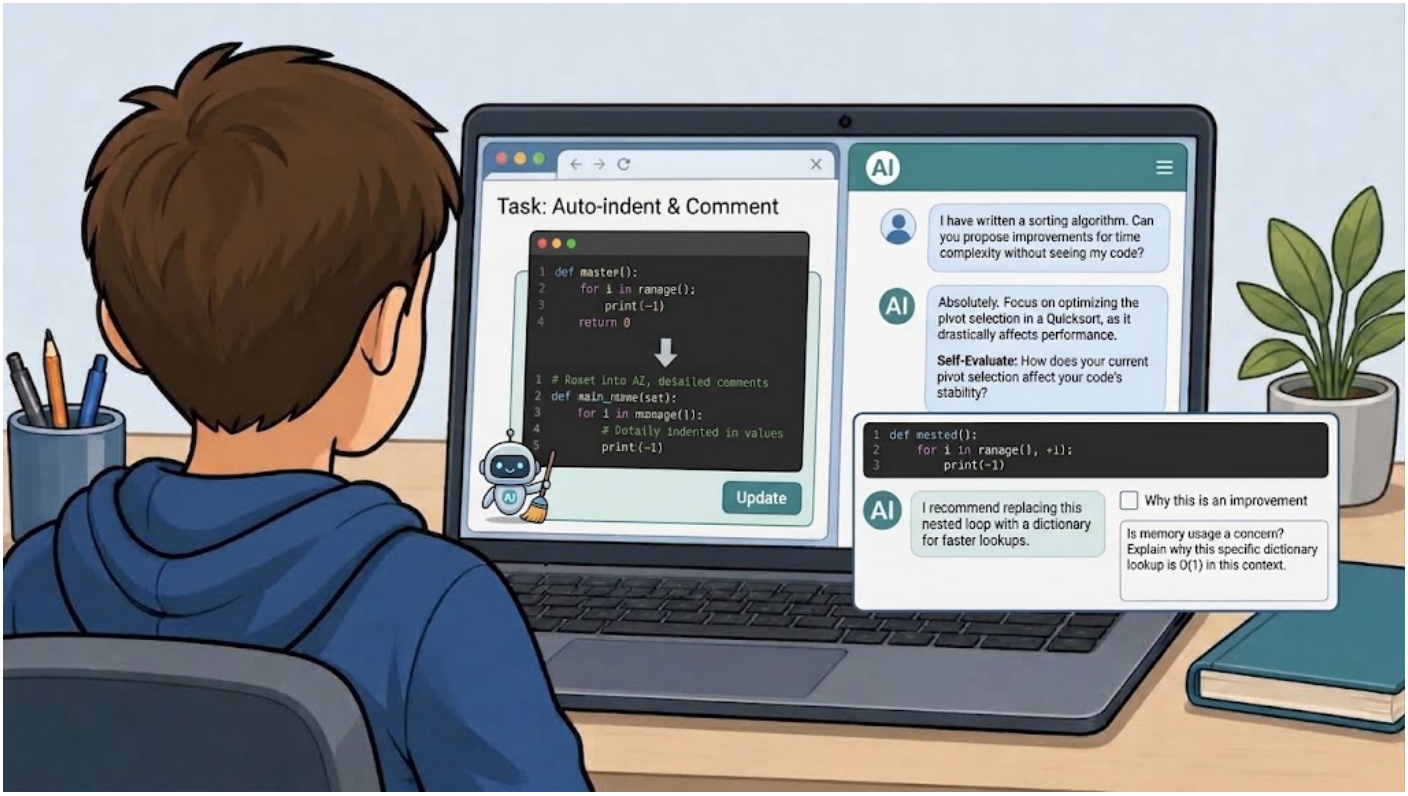
2.9 Programovanie: pomoc pri vybraných úlohách

Naučiť sa programovať bez toho, aby kód písala UI

Predmet: Programovanie · **Ročník:** 9 (vek 14 – 15) · **Trvanie:** 1 dvojhodina (90 min) · **UI:** Asistencia **S** UI

Kompetencie súvisiace s UI (rámec DUAL.AI.TEACHer) – úroveň učiteľa	**Predmetovo špecifické vzdelávacie ciele – úroveň študenta/žiaka**
AF-TL-2b *(Základy a aplikácie UI × Vyučovanie a učenie, úroveň 2 – Reflektívna implementácia)*: „Učitelia dokážu organizovať prvky hodiny, ktoré zapájajú používanie UI s jasnými pedagogickými rolami a stratégiami na overenie výstupných prvkov.“ **AF-FC-2b** *(Základy a aplikácie UI × Podpora digitálnej kompetencie (UI) žiakov, úroveň 2 – Reflektívna implementácia)*: „Učitelia dokážu poskytovať vedenie, ktoré pomáha žiakom rozvíjať praktické zručnosti v oblasti UI, pričom prispôsobujú úlohy predchádzajúcim vedomostiam žiakov a dostupným nástrojom UI.“	Na konci hodiny žiaci dokážu: • vedieť, ako používať UI na opakujúce sa či podporné úlohy pri programovaní, napríklad na odsadenie kódu alebo písanie komentárov • požiadať model UI, aby navrhol zlepšenia kódu bez toho, aby mu poskytli samotný kód, takže si žiaci môžu sami overiť svoje vedomosti • kriticky hodnotiť model UI, keď sa ho pýtajú, ako by daný kód optimalizoval

> **Hlavné posolstvo:** Ak majú žiaci hodnotiť výsledky, ktoré dá model, potrebujú poznať základy programovania. Preto musia žiaci v prvých fázach učenia sa programovania používať modely UI na podporné a opakujúce sa úlohy súvisiace s programovaním, no nie na tvorbu kódu. Model môžu tiež požiadať o možné zlepšenia kódu, aby si vyskúšali svoje programátorské schopnosti.



Obr. 1: „Naučiť sa programovať bez toho, aby kód písala UI“ (obrázok vytvorený pomocou UI, Google Gemini 3.6 Thinking)

Obsah

Vzdelávanie v programovaní sa mení v kontexte, v ktorom systémy UI dokážu generovať, vysvetľovať a kontrolovať kód. Hoci tieto nástroje prinášajú nové možnosti učenia, zavádzajú aj dôležitú vzdelávaciu výzvu: žiaci potrebujú dostatočné znalosti programovania, aby vedeli vyhodnotiť, či je návrh vygenerovaný UI správny, efektívny alebo vôbec relevantný pre danú úlohu. Bez základného porozumenia programátorským pojmom môžu žiaci prijímať nesprávne riešenia, prehliadať chyby alebo nedokázať rozpoznať lepšie alternatívy. Práve preto zostáva učenie sa základov programovania nevyhnutné aj vtedy, keď sú nástroje UI k dispozícii.

V úvodných kurzoch programovania by mala byť UI postavená do roly nástroja na podporu učenia, a nie zastupujúceho programátora. Žiaci môžu UI efektívne využívať na opakujúce sa alebo rutinné úlohy, ktoré neznižujú kognitívne úsilie spojené s učením sa programovať. Príkladom je formátovanie a odsadzovanie kódu, generovanie dokumentačných komentárov, vysvetľovanie chybových hlásení, sumarizovanie pravidiel syntaxe alebo pomoc žiakom pri pochopení účelu konkrétnych programátorských konštrukcií. Tieto spôsoby použitia umožňujú žiakom zamerať pozornosť na logiku, algoritmy a procesy riešenia problémov, ktoré sú pre programátorskú kompetenciu kľúčové.

Ďalším produktívnym využitím UI je kontrola kódu. Namiesto toho, aby žiaci požiadali model UI o napísanie riešenia, môžu si najprv napísať vlastný kód a potom model poprosiť, aby identifikoval možné zlepšenia. UI môže navrhnúť jasnejšie názvy premenných, lepšiu organizáciu kódu, menšiu

redundanciu, vyššiu čitateľnosť alebo alternatívne prístupy, ktoré žiak dokáže samostatne vyhodnotiť. UI sa tým mení na partnera pre spätnú väzbu a žiaci môžu porovnávať vlastné uvažovanie s vonkajšími návrhmi a posudzovať silné i slabé stránky svojich riešení.

Spätná väzba vygenerovaná UI by sa však nemala prijímať nekriticky. Modely môžu odporúčať zmeny, ktoré sú nepotrebné, neefektívne alebo vychádzajú z nesprávnych predpokladov o účele kódu. Hlavným cieľom tejto hodiny je preto rozvíjať schopnosť žiakov kriticky hodnotiť návrhy UI. Žiaci sa naučia pýtať sa modelu UI nielen na to, aké zlepšenia navrhuje, ale aj na to, prečo tieto zmeny považuje za prospešné. Porovnávaním argumentácie modelu s uznávanými programátorskými princípmi môžu žiaci posúdiť, či navrhovaná optimalizácia kód skutočne zlepšuje, alebo predstavuje len iné návrhové rozhodnutie.

Táto hodina pristupuje k UI ako k nástroju, ktorý podporuje reflexiu, sebahodnotenie a zvyšovanie kvality kódu, a nie ako k nástroju na generovanie kódu. Žiaci si precvičujú písanie vlastných programov, používanie UI na podporné úlohy, vyžiadanie spätnej väzby k svojej práci a kritickú analýzu kvality odpovedí, ktoré dostanú. Konečným cieľom je pomôcť žiakom stať sa lepšími programátormi aj poučenejšími používateľmi systémov UI.

Plán hodiny

Fáza	Čas	Aktivita
Úvod	10 min	Učiteľ predstaví dve krátke riešenia toho istého problému v kóde – jedno napísal žiak a druhé vygeneroval model UI. Žiaci diskutujú o tom, ktoré riešenie sa zdá jasnejšie, efektívnejšie alebo zrozumiteľnejšie.
Vstup	10 min	Učiteľ predstaví vhodné spôsoby využívania UI vo vzdelávaní v programovaní a zdôrazní rozdiel medzi podpornými úlohami, kontrolou kódu a generovaním kódu.
Skúmanie	15 min	Žiaci pracujú vo dvojiciach na malom programátorskom cvičení a určia, ktoré úlohy by bolo vhodné delegovať na asistenta s UI a ktoré majú zostať na nich.
Cyklus zlepšovania	35 min	Žiaci napíšu vlastné riešenie a prejdú 2 – 3 kolami postupu napíš → spýtaj sa → vyhodnoť → uprav, pričom UI používajú len na podporné úlohy a návrhy na zlepšenie.
Reflexia	20 min	Diskusia s celou triedou o užitočnosti a obmedzeniach spätnej väzby od UI so zameraním na to, ako žiaci určili, či boli návrhy hodnotné alebo nie.

Navádzajúce otázky: postupné pridávanie zložitosti

Žiaci prejdú počas 35-minútového cyklu zlepšovania toľkými kolami, koľko čas dovolí. Každé kolo má rovnaké štyri kroky: napíš → spýtaj sa → vyhodnoť → uprav

1. **Kolo 1 — Použitie UI na podporné úlohy**
   - Napíš: Samostatne vypracuj krátke programátorské cvičenie.
   - Spýtaj sa: Požiadaj UI o podporu pri úlohe, ktorá sa netýka samotného programovania, napríklad:
     - o vylepšenie odsadenia,
     - o vygenerovanie komentárov,
     - o vysvetlenie chybového hlásenia,
     - o opis toho, čo funkcia robí.
   - Vyhodnoť: Opisuje vysvetlenie od UI kód presne? Sú vygenerované komentáre užitočné a zrozumiteľné?
   - Uprav: Zapracuj užitočné zmeny, no dbaj na to, aby logika programu zostala výhradne tvojou vlastnou prácou.
2. **Kolo 2 — Vyžiadanie návrhov na zlepšenie**
   - Napíš: Prezri si svoj hotový program a nájdi časti, ktoré by sa dali zlepšiť.
   - Spýtaj sa: Vyžiadaj si návrhy na zlepšenie bez toho, aby UI kód prepisovala. Napríklad:

     > „Navrhni tri spôsoby, ako zlepšiť čitateľnosť a udržiavateľnosť tohto programu, ale neposkytuj náhradný kód.“

   - Vyhodnoť: Ktoré návrhy sa zdajú hodnotné? Ktoré návrhy by mali na program len malý vplyv?
   - Uprav: Zapracuj zlepšenia, s ktorými súhlasíš, a otestuj, či sa program stále správa správne.
2. **Kolo 3 — Kritické hodnotenie rád na optimalizáciu**
   - Napíš: Vyber časť svojho kódu, ktorá vykonáva opakujúcu sa úlohu alebo používa cykly.
   - Spýtaj sa: Vyžiadaj si návrhy na optimalizáciu a vyžaduj, aby UI svoje odporúčania zdôvodnila.

     Príklad promptu:

     > „Vysvetli, ako by sa dal tento kód optimalizovať a prečo by každá navrhovaná optimalizácia zlepšila výkon alebo čitateľnosť. Neposkytuj optimalizovaný kód.“

   - Vyhodnoť: Sú navrhované optimalizácie naozaj prospešné? Ktoré z nich sú podložené jasným zdôvodnením? Ktoré sa zdajú nepotrebné alebo ťažko odôvodniteľné?
   - Uprav: Rozhodni, ktoré odporúčania zapracuješ, a vysvetli svoje dôvody.

Otázky na reflexiu:

- Ktoré návrhy UI boli pri zlepšovaní tvojho kódu najužitočnejšie?
- Poskytla UI nejaké návrhy, s ktorými si nesúhlasil? Prečo?
- Ako si určil, či sa optimalizácia vyplatí?
- Kedy je UI počas programovania najužitočnejšia?

- Ktoré programátorské úlohy by mal žiak stále robiť sám?
- Prečo sú znalosti programovania potrebné na hodnotenie spätnej väzby vygenerovanej UI?
- Ako sa môžu žiaci pri učení sa programovania vyhnúť príliš veľkej závislosti od nástrojov UI?

Záverečná otázka (pre fázu reflexie):

Koľko znalostí programovania je potrebných na to, aby sa dalo určiť, či návrh UI program skutočne zlepšuje, a aké riziká vznikajú, keď sa používatelia spoliehajú na rady vygenerované UI bez toho, aby sami rozumeli kódu?

Materiály

- jeden notebook s programovacím prostredím pre každú dvojicu žiakov
- prístup k chatovému asistentovi s UI alebo k lokálne prevádzkovanému jazykovému modelu
- krátke programátorské cvičenie primerané úrovni skúseností žiakov
- pracovný list podľa cyklu napíš → spýtaj sa → vyhodnoť → uprav
- otázky na reflexiu zamerané na gramotnosť v oblasti UI a kritické hodnotenie
- voliteľné príklady kódu ilustrujúce dobrú a slabú programátorskú praxu

---

Revision #3
Created 2026-09-17 21:47:53 UTC by Alexander
Updated 2026-10-29 14:16:12 UTC by Jannik