

SLOVENSKÁ TECHNICKÁ UNIVERZITA V BRATISLAVE

NÁVRH UČEBNÉHO TEXTU V PREDMETE INFORMATIKA

ZÁVEREČNÁ PRÁCA  
DOPLŇUJÚCEHO PEDAGOGICKÉHO ŠTÚDIA



SLOVENSKÁ TECHNICKÁ UNIVERZITA V BRATISLAVE  
ÚSTAV MANAŽMENTU

NÁVRH UČEBNÉHO TEXTU V PREDMETE INFORMATIKA

ZÁVEREČNÁ PRÁCA  
DOPLŇUJÚCEHO PEDAGOGICKÉHO ŠTÚDIA

Vedúci záverečnej práce: doc. Ing. Gabriela Pavlendová, PhD.

Bratislava, 2023

Bc. Miroslav Hájek



## Pod'akovanie

Chcel by som vyjadriť vďačnosť predovšetkým svojim niekdajším učiteľom na gymnáziu, z ktorých niektorí sa stali kolegovia. Moje srdečné ďakujem patrí učiteľke matematiky PaedDr. Jane Homolovej, učiteľke fyziky Mgr. Jane Hornychovej, učiteľke informatiky Ing. Darine Runovej a učiteľke geografie RNDr. Helene Bajerovej. Podarilo sa im ukázať mi, že vyučovať technické a prírodovedné predmety sa dá s obrovským entuziazmom a citom pre kreatívne prepojenie učiva s bežným životom. Poskytli mi príležitosti vymýšľať a overovať rôznorodé problémové úlohy. Spočiatku v projekte Geovedy pre každého, potom pri spoluorganizovaní programátorského krúžku pre rovesníkov a teraz pri výučbe v role učiteľa. Tak bola preverená aj táto zbierka úloh.



## Čestné prehlásenie

Čestne vyhlasujem, že som túto prácu vypracoval samostatne, na základe konzultácií a s použitím uvedenej literatúry.

V Bratislave, 2023

.....

Bc. Miroslav Hájek





## Abstrakt

Závěrečná práce sa venuje tvorbe učebníc so zameraním na organizáciu a na obsahovú štruktúru systému úloh podľa náročnosti, tak aby boli všestranne podporné didaktické funkcie učebného textu. Za týmto účelom sa predstavuje inovatívna zbierka úloh z programovania so vzorovými riešeniami v jazyku Python pre stredné školy vo všeobecnovzdelávacom vyučovanom predmete informatika. Problémové cvičenia sú po vzore súťažných úloh zasadené do kontextu krátkych príbehov, ktoré motivujú žiaka k užitočnosti konkrétnych algoritmov a programov. Aplikačné oblasti sú volené s prihliadnutím na rozvoj medzipredmetových vzťahov. Široká dostupnosť zbierky je podporená prezentáciou v hypertextovom priestore webu. Kolaboratívna rozširiteľnosť sa umožňuje vypracovanou metodikou stanovujúcou pevnú predlohu úlohy, požiadavkami na kvantitatívne vlastnosti znenia zadaní a zaradením úloh do zbierky podľa témy, funkcie a poznávacej úrovne. Výsledkom je sada problémových úloh v súlade so štátnym vzdelávacím programom a cieľovými požiadavkami na maturitnú skúšku, ktorá má aktivizovať žiakov v problémovom vyučovaní a dopomáha k samostatnej domácej príprave.

**Kľúčové slová:** problémové vyučovanie, učebnice, zbierka úloh, informatika, základy programovania



# Abstract

The final thesis is aimed at the textbook design with a focus on the organization and content structure of the system of problems according to difficulty so that the didactic functions of the teaching text are supported comprehensively. For this purpose, a collection of programming problems with sample solutions in the Python language is presented for secondary schools in the general education subject of computer science. Following the model of competition problems, problem exercises are set in the context of short stories, which motivate the student to the usefulness of specific algorithms and programs. Application areas are chosen to develop interdisciplinary relationships. The wide accessibility of the textbook is supported by its presentation in the hypertext web environment. Collaborative extensibility is made possible by the developed methodology establishing a fixed template of the problem, requirements for quantitative properties of problem wording, and insertion of problems in the textbook according to topic, function, and cognitive level. The result is a set of problems in the scope given by the state educational program and the target requirements for the matura exam, which is supposed to engage students in problem-based learning and help them with individual home preparation.

**Keywords:** problem-based learning, textbooks, collection of problems, computer science, programming basics



# Obsah

<b>1</b>	<b>Úvod</b>	<b>1</b>
<b>2</b>	<b>Tvorba edukačných materiálov</b>	<b>3</b>
2.1	Učebnica . . . . .	3
2.1.1	Didaktické funkcie učebnice . . . . .	4
2.1.2	Prvky učebnice . . . . .	5
2.1.3	Multimediálne prostredie . . . . .	6
2.1.4	Skvalitňovanie učebného textu . . . . .	7
2.2	Systém úloh v zbierke . . . . .	9
2.2.1	Psychologické východiská . . . . .	9
2.2.2	Klasifikácia vlastností úlohy . . . . .	10
2.2.3	Preformulovanie úloh . . . . .	12
2.3	Vzdelávacie štandardy v informatike . . . . .	13
2.3.1	Algoritmické riešenie problémov . . . . .	13
2.3.2	Existujúce učebnice a zbierky úloh . . . . .	14
<b>3</b>	<b>Cieľ a metodika práce</b>	<b>19</b>
<b>4</b>	<b>Výsledky práce</b>	<b>21</b>
4.1	Zbierka úloh . . . . .	21
4.1.1	Premenné . . . . .	21
4.1.2	Podmienky . . . . .	27
4.1.3	Cykly . . . . .	30
4.1.4	Náhodné čísla . . . . .	34
4.1.5	Reťazce a zoznamy . . . . .	36
4.1.6	Súbory . . . . .	39
4.1.7	Funkcie . . . . .	42
4.2	Vzorové riešenia . . . . .	45

4.3	Elektronická učebnica . . . . .	47
4.4	Diskusia . . . . .	48
4.4.1	Číselný opis textov zadaní . . . . .	49
4.4.2	Predloha štruktúry úlohy . . . . .	50
4.4.3	Didaktické funkcie úloh . . . . .	51
4.4.4	Medzipredmetové vzťahy . . . . .	53
<b>5</b>	<b>Záver</b>	<b>55</b>
	<b>Bibliografia</b>	<b>57</b>
<b>A</b>	<b>Kompletné riešenia úloh</b>	
<b>B</b>	<b>Zaradenie úloh do zbierky</b>	
<b>C</b>	<b>Výpočet skóre čitateľnosti</b>	



# 1 Úvod

Rýchly vývoj informatiky ako vedy za posledné desaťročia, rapídny technologický rozvoj vedúci k zvýšeniu dostupnosti prostriedkov výpočtovej techniky a nástup inovatívnych prístupov vo výučbe umocňuje nedostatok moderných kvalitných vzdelávacích materiálov pre stredné školy. Na vzdelávanie informatiky na úrovni vyššieho sekundárneho vzdelania nevplýva ani tak posun v podstatných princípoch odboru, ale skôr výskyt nových informačných a komunikačných technológií prinášajúcich predtým nepoznané výzvy do spoločnosti. Nemenej podstatný dopad majú schopnosti nástrojov, čiže vlastností softvérového vybavenia a jeho používateľského rozhrania

Pre niektoré oblasti, ako sú kybernetická bezpečnosť alebo programovanie vychádzajú v súčasnosti už aktuálne učebnice, ale častokrát sú školám nedostupné najmä z ekonomických dôvodov. Poznatky v tlačенých učebniciach zároveň zvyknú rýchlo zastarávať, preto sa preferujú elektronické knihy. E-knihy majú zasa z pohľadu vyučovacieho procesu úskalia ohľadom ich prístupnosti a prevládajúceho tradicionalizmu.

Význačné zmeny sa udiali v programovacích jazykoch ako nástrojoch na formálny zápis algoritmov. Ovplyvnili výber prostredí na programovanie a jazykov pre použitie vo vzdelávaní a postupnosť v osvojovaní prvkov jazyka. Nové verzie neustále prinášajú úpravy syntaxe a údajových štruktúr.

V záujme udržania kroku s najnovšími relevantnými poznatkami v odbore a stavom technológií, sú učitelia často nútení pripravovať si vlastné učebné texty. Napomáhajú im k tomu internetové zdroje a e-learningové knižnice. Ponechávajú však časovo náročný výber adekvátnych článkov a vyváženú skladbu cvičení na kreativite učiteľa. Na prvý pohľad sa to môže javiť prospešné pre individualizáciu výučby. Realizácia nebýva nikdy ideálna, tak aby žiakom poskytla rozmanité úlohy na samostatnú domácu prípravu.

Na vyučovacích hodinách by mala dobrá učebnica vhodne podnecovať a podporovať problémové vyučovanie, ktoré aktivizuje žiakov k hlbšiemu ovládaniu preberanej témy. Okrem iných časových a priestorových obmedzení vplývajúcich na výber učebnej metódy, vedie učiteľa nedostupnosť usporiadaného učebného textu na sprevádzanie



učivom, hlavne k využitiu frontálneho výkladu.

V svojpomocnej tvorbe učebníc všeobecnovzdelávacieho učebného predmetu informatika sa zameriavame na vzdelávací štandard algoritmické riešenie problémov podľa štátneho vzdelávacieho programu. V snahe preniesť úsilie v triede pri osvojovaní učiva vo väčšej miere na žiaka, vychádzame z potreby zostavenia zbierky úloh z programovania pre stredné školy k použitiu v škole aj na doma. Nevyhnutné zložky hodné komplexného posúdenia sú obsahová a formálna rovina.

Na obsah zbierky kladieme nároky na vyvážené pokrytie dôležitých typov úloh na viacerých kognitívnych úrovniach v súlade s princípmi formulácie úlohy aj systematickým usporiadaním úloh. Spôsob začlenenia riešení do zbierky a ich fáza konfrontovania s postupmi žiakov je tiež neoddeliteľnou súčasťou učebného textu ako celku. Na predchádzanie straty aktuálnosti textov budeme požadovať čo najväčšiu nezávislosť úloh na programovacím prostredí a jazyku.

Po formálnej stránke máme záujem navrhnúť „živú učebnicu“, kde by mohli učitelia v online prostredí postupne dopĺňať nové úlohy do jednotlivých častí zbierky. Na tento účel prispôbujeme existujúcu metodiku hodnotenia náročnosti textu a zaraďovania úloh do systému úloh podľa príslušných kritérií. Prihliada sa pritom na obsahové zameranie v rámci tematických okruhov podľa znenia zadania. Namiesto opakujúcich sa typov cvičení sa nájde náhrada vhodným preformulovaním.

V hlavnej časti práce preskúmame teóriu tvorby učebníc s ohľadom na funkcie, skvalitňovanie učebného textu a na nástrahy pri presune do elektronickej podoby. Ďalej sa venujeme zostrojeniu adekvátneho a rozšíriteľného systému úloh podľa zbierok príkladov z matematiky. Nasleduje prehľad a rozbor súčasných učebníc programovania a zbierok úloh. V praktickej časti predstavíme konkrétne úlohy so vzorovými riešeniami a hodnotením zaradenia či náročnosti úloh. Nakoniec prediskutujeme typické vlastnosti textu úloh po obsahovej a grafickej stránke.

## 2 Tvorba edukačných materiálov

Súčasný stav problematiky tvorby edukačných materiálov obsahuje zákonitosti stavby a funkcií učebnice, odporúčania pri písaní zrozumiteľných učebných textov a rozvrhovaní didakticky správneho systému úloh v zbierke. Tiež porovnáme doterajšie učebnice programovania pre stredné školy navzájom a vzhľadom na štátny vzdelávací štandard.

### 2.1 Učebnica

Základným prameňom poznatkov a nositeľom obsahu vzdelávania je učebnica, ktorá patrí medzi čelných predstaviteľov pedagogických textov. Predstavuje jadro zoskupujúce okolo seba ostatné učebné prostriedky (Zujev, 1986). V medziach učebných osnov vymedzuje obsah základného učiva s doplnením o rozširujúce učivo, pričom rozsah osnovy učebnice nemusí byť totožný s učebnými osnovami (Mladý, 1988).

Učebnica pomáha žiakom s osvojením si obsahu učiva, čím podporuje všetky súvisiace čiastkové činnosti: precvičovanie, opakovanie, systematizáciu a integráciu. V edukačnom procese pôsobí učebnica aj výchovne, čím vplýva na formovanie postojov, motívov a záujmov. Odlišuje sa v tom od iných kníh a textov, pretože má priamu späťosť so získavaním a spracovaním faktov, pojmov a vzťahov žiakmi. Efektívne tak smeruje dosiahnutie výchovno-vzdelávacích cieľov vyučovacieho predmetu (Gavora, 1992). Učebný program žiaka a vyučovací program učiteľa je ideálne v učebnici podchytený a odráža sa do scenáru učebného a vyučovacieho procesu. Hlavná časť učebnice predstavuje súbor úloh určených na aktívne riešenie (Pavlovkin et al., 1989).

Podľa školského zákona sa učebnica spolu s učebným textom a pracovným zošitom zaraďuje medzi edukačné publikácie. Na vzdelávanie sa používajú publikácie schválené ministerstvom školstva alebo zodpovedajúce princípom a cieľom výchovy a vzdelávania (NRSR, 2023). Princípy súvisiace s vlastnými vzdelávacími materiálmi sú v duchu rovnoprávnosti, rovnocennosti, zodpovednosti, tolerance a vyváženého rozvoja osobnosti a zdokonaľovania vzdelávania podľa výsledkov výskumu a vývoja.

### 2.1.1 Didaktické funkcie učebnice

V učebnici pôsobí viacero naviazaných a prelínajúcich sa vlastností vystupujúcich vo výchovno-vzdelávacom procese (Zujev, 1986), ktoré sú tu opísané bez stanoveného poradia dôležitosti:

- **Informačná funkcia:** sa sústreďuje na vymedzenie povinného rozsahu informácií pri štúdiu nevyhnutných na zapamätanie.
- **Transformačná funkcia:** spočíva v didaktickom transfere poznatkov vedného odboru na obsah učiva v zrozumiteľnej a pútavej podobe. Zároveň berie ohľad na vekové a kultúrne osobitosti žiakov. Nabáda na výber vzdelávacích metód a uľahčuje aktivizáciu žiaka pri cvičeniach a úlohách prieskumného charakteru. Pri transformácii znalostí sa hľadá sa aj na potreby profesijného života a spoločenského očakávania od absolventov.
- **Systematizačná funkcia:** pri objasňovaní učiva zabezpečuje následnosť poznatkov, postupný nárast náročnosti a vedie k metódam vedeckej systematizácie.
- **Usmerňujúca funkcia:** slúži k upevňovaniu vedomostí, napomáha v orientácii sa v nich a zapojením ich do praktických druhov činností. Vyžaduje sprevádzanie navrhnutými aktivitami pod vedením učiteľa.
- **Motivačná funkcia:** pobáda túžbu a schopnosti žiakov na samostatné získavanie vedomostí.
- **Integračná funkcia:** ucelene spája poznatky žiakov nadobudnuté z ich rozličných činností.
- **Koordinačná funkcia:** zapája ku vzťahu k študovanému predmetu informácie z masovokomunikačných prostriedkov.
- **Výchovná funkcia:** súčasne tiež rozvíjajúca funkcia. Spočíva v zladenom formovaní črt osobnosti žiaka.

Uvedené didaktické funkcie zabezpečujú komplexné pôsobenie učebnice na rozvoj kognitívnych a afektívnych schopností žiaka. Pri ich prepojení v učebnom texte dochádza nielen k nadobudnutiu nevyhnutných vedomostí a zručností na zvládnutie vyučovacieho predmetu, ale aj na rozvoj kľúčových kompetencií a medzi nimi pripravenosť k ďalšiemu vzdelávaniu sa.

### 2.1.2 Prvky učebnice

Aby učebnica plnohodnotne naplňala svoje mnohé poslania, je poskladaná z **prvkov textového i mimotextového charakteru**, ktoré podnecujú aktívne kognitívne procesy a umožňujú zapojenie zvoleného učebného štýlu čitateľa. Zaradenie súčastí učebnice do členenia jej podsystémov nie je striktné, ale riadi sa dominantnou funkciou danej časti (Zujev, 1986).

Text v učebnici je súhrn viacerých viet, ktoré sú prostriedkom na odovzdávanie informácií žiakom podľa komunikačného zámeru autora. Z povahy súvislého písaného jazykového prejavu sa vyznačuje kohéziou a koherenciou. Kohézia je súdržnosť textu na úrovni vzájomnej nadväznosti medzi vetami za použitia gramatických, lexikálnych a grafických jazykových prostriedkov. Najčastejšie sa uplatňujú gramatická zhoda medzi nadradeným a podradeným slovným druhom alebo vetným členom, opakovanie výrazu ďalej v texte, synonymá, a interpunkčné znamienka. Koherencia je zase tematická spojitosť textu, keď sa z hlavnej rozvíjajú vedľajšie myšlienky (Gavora, 1992).

Podľa úlohy, ktorú zohráva text v predstavovaní učebnej látky sa rozlišuje **základný, doplnujúci a vysvetľujúci text**. Základný text je určený ako povinný na osvojenie pre zvládnutie problematiky. Podľa typu činností motivovanej základným textom sú povšimnuté teoretické poznávacie texty, považované tiež za výkladovú zložku zastávajúcu informačnú funkciu vysvetľovania a komentovania nového učiva. Naproti tomu u inštrumentálno-praktických textov a nevýkladovej zložky prevláda transformačná funkcia premietajúca sa do otázok, úloh a cvičení. Doplnujúci text prehľbuje rozsah učebných osnov dodatočnou argumentáciou vplývajúcej na rozumovú a emočnú stránku. Regulovanie poznávacej činnosti má na starosti vysvetľujúci text, ktorý dáva základný text do súvislostí (Zujev, 1986).

Mimotextové zložky nachádzajúce sa v učebnici sa rozlišujú na **aparát organizácie osvojovania, ilustračný materiál, a orientačný aparát**. Na organizáciu osvojovania slúžia prehľadové tabuľky, otázky a úlohy spolu s odpoveďami, ktoré v závislosti od kontextu môžu spadať aj do základného textu. Ilustrácie prevažne graficky dotvárajú textovú zložku, s ktorou sú vo vzťahovej rovine, buď nadradenosti ako vedúce ilustrácie, rovnocennosti, alebo podriadenosti ako doplnkové ilustrácie. Súvislosť medzi textom a ilustráciou sa pozná podľa toho, či text opisuje ilustráciu, vtedy je text podriadený, alebo ilustrácia slúži na dokreslenie situácie. Typickými príkladmi

ilustrácií sú obrázky, schémy, plány, diagramy, grafiky, mapy. Orientačný aparát slúži na zdôraznenie slovných spojení alebo myšlienok cez tlačové zvýraznenia a symbolické značenie. Opakuje tiež prvky z hlavnej časti v zhrnutnej podobe na navigovanie v knihe, v čom spočíva úloha predhovoru, obsahu, a registrov (Zujev, 1986).

Tradičná redakčná výroba učebnice dbá na ustálené metódy a organizovanú spoluprácu pre kontrolu správnosti obsahu po odbornej stránke. Redakcia dohliada na gramatickú, pravopisnú a štylistickú úpravu (Mladý, 1988). Zasadzuje sa o koordináciu činnosti autorských kolektívov, tak aby umožnila zosúladiť všetky dôležité prvky učebnice najmä súhrnu textovej a grafickej časti. Osvedčené pracovné postupy redakcie zabezpečujú zahrnutie podstatných zložiek učebnice. Postup sa začína sa vypracovaním osnovy učebnice, príprave materiálov a podkladov. Nasleduje príprava rukopisu a obrazových predlôh, v čase vymedzenom stanoveným harmonogramom, ktoré následne prechádzajú korektúrou. Celkové snaženie je zavŕšené vydaním učebnice a získaním doložky od ministerstva školstva podľa osobitého predpisu. Tvorba vzdelávacích materiálov priamo učiteľmi nie je až tak rigidná, ale redakčné techniky navádzajú aspoň na zmysluplnú organizáciu práce.

### 2.1.3 Multimediálne prostredie

Postavenie výuky informatiky okolo počítačov a súvisiaceho prídavného vybavenia vedie k prirodzenej snahe uspokojovať edukačné materiály naskytajúcim sa podmienkam. Učebné texty tým môžu zužitkovať príležitosti na obohatenie ich obsahu multimédiami a hypertextovými prepojeniami. Princípy uplatňujúce sa pri tvorbe klasických učebníc sa nevyhnutne prenášajú aj na tvorbu multimediálnych učebníc, pretože rovnako zostávajú publikáciami uspokojenými ku didaktickej komunikácii (Krotký, 2015).

Množstvo existujúcich učebníc prechádza do online prostredia zo svojej pôvodne knižnej úpravy na zvýšenie ich atraktívnosti a pohodlia pri prístupe k nim. Ani vznik učebnice určenej primárne pre elektronické médium však ešte nezaručuje využitie plného potenciálu na skĺbenie inovatívnych vyučovacích metód a ponúknutých technických vymožeností. Učebnice sa preto odlišujú podľa miery ich splynutia s digitálnym prostredím na **jednoduché, komplexné a pokročilé učebnice** (Krotký, 2015).

Jednoduché učebnice sú elektronickým obrazom svojich papierových vzorov bez uplatnenia akýkoľvek nových rozširujúcich možností. Komplexné učebnice získame za-

komponovaním multimediálnych prvkov, zastúpených prevažne zvukmi, obrázkami, animáciami, videom, a vložením hypertextových odkazov smerujúcich dovnútra vlastného obsahu, na externé webové portály a ďalší multimediálny obsah. Pokročilé učebnice navyše pozostávajú z interaktívnych prvkov aktívne prispôsobujúcich tok informácií. Interakcia sa odohráva cez tlačidlá, posuvníky, kontextový pomocník, a príbuzné ovládanie. Nadstavbou pokročilej učebnice je edukačný softvér.

**Interaktívne personalizované úlohy** slúžia na obohatenie osvojenia vedomostí z multimediálnej učebnice. Snaha vedie k zníženiu kognitívnej záťaže pri návrhu používateľských rozhraní, tak aby sa žiak mohol sústrediť výhradne na osvojované učivo, než na ťažkosti s komplikovanými krokmi na dosiahnutie vytýčených zámerov vo virtuálnom priestore. Teórie venujúce sa tomuto problému súvisia s obmedzeniami kapacity krátkodobej pamäte.

**Teória kognitívnej záťaže** odporúča eliminovanie vonkajšej (*extraneous*) kognitívnej záťaže cez zjednodušenie kompozície zobrazovaných prvkov. Vonkajšia kognitívna záťaž zaberá miesto vnútornej (*intrinsic*) a konceptuálnej (*germane*) záťaži, ktoré sú potrebné na riešenie samotnej úlohy (Uherčík, 2012). **Teória duálneho kódovania** hovorí, že verbálne a neverbálne podnety sú v pamäti kódované zvlášť. Tým sa zvyšuje počet položiek v krátkodobej pamäti, pokiaľ pochádzajú z odlišného zdroja (Mishra et al., 2005).

#### 2.1.4 Skvalitňovanie učebného textu

Vylepšenia v učebných textoch sa uskutočňujú na základe teoretických východísk z porozumeniu textu pri **čítaní ako psycholingvistickej činnosti**. Na úspešné odhalenie komunikačného zámeru pozná recipient vzťahy medzi objektívnou realitou a na ňu odkazujúcimi prvkami textu, medzi jednotlivými prvkami textu a medzi textom a doterajšími znalosťami prijímateľa (Gavora, 1992).

Operáciou elaborácie čitateľ nachádza asociácie v prečítanom texte s už nadobudnutými sémantickými epizodickými znalosťami a vizuálnymi predstavami, pokiaľ jestvuje také spojenie. Inferencia umožňuje doplnenie zamlčaných informácií v texte, ktoré vyplývajú z opísaných príčinnno-dôsledkových súvislostí (Gavora, 1992). Počas zvnútorňovania edukačného textu dochádza k postupu od jeho porozumenia bez vzťahu k iným textom, cez prevod na parafrázy a symbolický zápis, cez interpretáciu vnášajú-

cej odlišný pohľad na prečítané, až ku extrapolácii inovatívnych záverov a schopnosti predpovedať dôsledky (Gavora, 1992).

Na objasnenie nových konceptov je teda prospešné, ak sú uvádzané východiskové situácie povedomé a autorova predstava sa zhoduje s čitateľovou. Čítaním môže nastať neporozumenie v tzv. **mikroštruktúre textu**. Do mikroštruktúry patria slová, vety, vzťahy medzi vetami, či štruktúra textu. Neznámym slovám dokážeme predchádzať ich vhodným výberom usúdenej z náročnosti pojmu. Na to slúžia taxonomické normy zachytávajúce typickosť pojmu, tým že ho zaradia pod názov nadradenej kategórie. Zložité vety odkazujúce sa vedľajšou vetou na vzdialené slová a včlenené prívlastkové vety by mali byť radšej rozdelené na dve vety. Pozornosť pri rozdeľovaní treba venovať neopomenutiu podstatných spájajúcich slov.

**Zložitosť textu** sa opisuje kvantitatívnymi charakteristikami, ktoré sa prejavujú v čitateľnosti a náročnosti textu. **Čitateľnosť** sa zvykne merať dĺžkou viet alebo výskytom neobvyklých slov. Jednou z mnohých mier čitateľnosti je Gunning *Fog index*, v prijateľnom skóre pre stredné školy do 14 (Vzorec 2.1) (Drahošová, 2014). **Náročnosť** textu vychádza z lexikálnych a syntaktických faktorov, ktoré pokladajú na škálu zložitosti toho čo je povedané a akým spôsobom je to zapísané. *Průchová modifikácia Nestlerovej metody* zisťuje obtiažnosť na vzorkách výkladového textu učebnice zohľadnením syntaktickej a sémantickej náročnosti. Pod 20 bodov sa jedná o text nízkej obtiažnosti a nad 60 bodov ide o texty vysokej obtiažnosti. (Vzorec 2.2) (Drahošová, 2014). Textu vyjadrujeme tiež inferenčnú záťaž, čiže nutnosť vyvodzovania vzťahov čitateľom. Znižuje sa umiestnením podobných myšlienok za sebou (Pavlovkin et al., 1989).

$$0.4 \cdot \left( \left( \frac{\sum \text{slova}}{\sum \text{vety}} \right) + \left( \frac{\sum \text{slova nad 2 slabiky}}{\sum \text{slova}} \cdot 100 \right) \right) \quad (2.1)$$

$$0.1 \cdot \left( \frac{\sum \text{slova}}{\sum \text{vety}} \right) \cdot \left( \frac{\sum \text{slova}}{\sum \text{slovesa}} \right) + \left( \frac{\sum \text{pojmy}}{\sum \text{slova}} \right) \cdot \left( \frac{\sum P_1 + 3 \sum P_2 + 2 \sum P_3 + 2 \sum P_4 + \sum P_4}{\sum \text{slova}} \right) \quad (2.2)$$

Ku kvalite textov učebnice prispieva aj **makroštruktúra textu**, ktorá rozčleňuje tematické okruhy na kapitoly a tie na texty. Na poskytnutie nadhľadu slúžia typografické zvýraznenia hrubým písmom alebo oddeľovacím prázdny priestorom, nadpisy rôznych veľkostí na rozlíšenie tém a podtém, kľúčové vety vyjadrujúce hlavnú

myšlienku, a uvádzajúce či rekapitulujúce otázky pobádajúce čitateľa na aktívne prijímanie materiálu (Pavlovkin et al., 1989). Nemenej znateľná pri čítaní je primerane jednoduchá grafická úprava, ktorá neodpúta pozornosť od obsahu. Prehľadnosť sa zlepšuje nastavením ľahko čitateľného písma s veľkosťou odrážajúcou hierarchiu celkov, riadkovaním do bloku, a zalamovaním príkladov v celku na jednu stranu (Mladý, 1988).

Na hodnotenie kvality učebníc sa uplatňujú experimentálne, expertné a štatistické metódy. V autentickom školskom prostredí môžeme experimentálne overovať a porovnávať navrhovanú učebnicu so staršou zaužívanou. Pozorovatelia učebnice ako sú experti, učitelia a žiaci hodnotia rozličné vlastnosti, s ktorými prichádzajú do kontaktu, napríklad primeranosť, metodické spracovanie, zaujímavosť, zložitosť. Štatisticky sa kvantifikuje rozsah textu určený na vyučovaciu jednotku, čitateľnosť a náročnosť textu.

Výskum Drahošovej sumarizuje techniky pre pedagogickú prax na zlepšovanie zrozumiteľnosti učebného textu. Ohľadom výberu slov odporúča uprednostniť bežné slová, opisateľné pojmy a aktívne slovesá, zároveň sa vyhýbať nepotrebným slovám. V rovine štylistiky by sa malo písať s priblížením sa bežne hovorenej reči, v jednoduchých celkoch a krátkych vetách. Myšlienky textu vyjadrovať adresne a presvedčivo s opieraním sa o skúsenosti čitateľa (Drahošová, 2014).

## 2.2 Systém úloh v zbierke

Skupina úloh sa označuje za systém úloh, keď plní konkrétnu didaktickú funkciu v súlade s učebnými cieľmi, štruktúrou poznávacieho procesu a podmienkami učebného procesu (Mindáková, 2008). Otázky na precvičovania učiva sa vyznačujú špecifikami oproti výkladovému textu, prevažne tým že ich riešenie bezpochyby vyžaduje aktívnu činnosť žiaka na rôznych úrovniach myslenia. Učiteľ v tomto štádiu pozoruje postup žiakov pri vypracovaní úloh a na základe ich vonkajších prejavov posudzuje a usmerňuje ich činnosť k želanému cieľu. Kritické je stanoviť následnosť a hierarchiu úloh, tak aby umožňovali kontinuálny rozvoj žiaka.

### 2.2.1 Psychologické východiská

Návrh systému úloh sa zapodieva s otázkami žiackej motivácie riešenia úloh, diferenciácie úloh vzhľadom na individuálne osobitosti žiakov, vhodného zoradenia úloh od



jednoduchších k zložitejším a spôsobu merania stupňa zvládnutia učiva. Nedá sa predpokladať, že všetky témy budú samy osebe atraktívne. Dieťa sa však aspoň odhodlá k takým úlohám, ktoré sa domnieva že prekoná bez pociťovaných ťažkostí.

Úspech prirodzene motivuje na skúšanie väčších výziev. Neprijemné skúsenosti a zlyhanie žiakov postupne odrádzajú. Výber a zoradenie úloh má zaručiť zážitky úspechu. Primeraný cvičebný materiál zodpovedá poznávaciemu potenciálu dieťaťa vo **vekových osobitostiach, individuálnych odlišnostiach a predchádzajúcich skúsenostiach**. Pokrok v psychickom vývine vyšších schopností dosiahneme zaradením nielen veku primeraným problémov, ale aj rozvíjajúcich na rozšírenie zóny najbližšieho vývinu podľa Vygotského. Z hľadiska individuálnych predpokladov na riešenie úlohy sa treba zamýšľať nad optimálnymi poznávacím štýlom, celkovou úrovňou schopností a profilom schopností na splnenie konkrétneho zadania. V zbierkach sa preto ponúkajú úlohy troch úrovní náročnosti: **menej náročné, stredné, vysoko náročné** (Pavlovkin et al., 1989).

Skupinu menej náročných úloh vedia riešiť priemerní žiaci v nižšom ročníku, teda sú určené na opakovanie a pre žiakov s pomalším tempom vývinu či nižšou poznávacou kapacitou. Stredne náročné úlohy s najväčšou početnosťou sú pochopiteľné pre priemerných žiakov v danom ročníku. Žiaci s vyššou poznávacou kapacitou dokážu prejsť vysoko náročnými úlohami, ktoré sú určené pre priemerných vo vyššom ročníku.

Úlohy by mali byť prispôsobené okrem schopností žiaka aj vedomostiam a spôsobilostiam. Získavanie nových schopností prechádza od nadobudnutia vedomostí v podobe pojmov a schém, cez osvojovanie spôsobilostí v priebehu čoraz vyladenejšieho motorického, senzomotorického a psychického cvičenia, až k rozvíjaniu myslenia spojeného so stratégiami formulácie problému a plánovania riešenia (Pavlovkin et al., 1989).

Efektívneho učenia celkove dosiahneme podľa Skinnera, keď sú známe konkrétne ciele výchovy a vzdelávania, žiakom je umožnené postupovať vlastným tempom nezávisle na ostatných, a okamžitou spätnou väzbou s odhalením správnej odpovede (Pavlovkin et al., 1989).

### 2.2.2 Klasifikácia vlastností úlohy

Naplnenie tematického celku náročnosťou odstupňovanými úlohami s rozmanitými didaktickými funkciami a zapojením kognitívnych funkcií naprieč úrovňami myšlienko-

vých operácií sa dá skontrolovať cez špecifikáciu vlastností konkrétnej úlohy. Od charakteristík úlohy sa odvíja jej zaradenie do zbierky a sú to (s príkladmi z matematiky) (Mindáková, 2008):

- **Téma:** názov tematického celku vo vyučovacom predmete (*napr. Funkcia*).
- **Podtéma:** téma sa rozdeľuje na viaceré časti (*napr. Lineárna funkcia, ...*)
- **Element:** pojmy, vzťahy a procesy podľa obsahového štandardu. Rozsiahlejšie elementy môžu vystupovať ako podtémy (*napr. Pytagorova veta*)
- **Funkcia:** didaktické požiadavky na poznávací proces. Prípustné je ak úloha napĺňa niekoľko didaktických funkcií, napr. slovná úloha na upevnenie učiva s aplikáciou poznatkov mimo matematiky. Úlohy na základe didaktickej funkcie podľa D. Švedu sú (Šveda, 1992):
  - a) úlohy na motiváciu učebnopožívacej činnosti žiakov
  - b) úlohy na aktualizáciu skôr osvojeného učiva
  - c) prípravné úlohy predchádzajúce vysloveniu definície pojmu a riešeniu základných úloh
  - d) úlohy na osvojenie definície pojmu, formulácie vety a postupu riešenia
  - e) úlohy na upevňovanie učiva
  - f) úlohy na aplikáciu učiva mimo informatiky
  - g) úlohy na aplikáciu učiva vo vnútri informatiky
  - h) úlohy propedeutického charakteru k nasledujúcim elementom učiva v tematickom celku
  - i) úlohy na opakovanie a systemizáciu
- **Úroveň:** úloha rozvíja zároveň všetky nižšie úrovne poznávacích procesov, preto sa označuje iba najvyššou. Poznávacie procesy podľa M. Zelina sú (Zelina, 1990):
  - a) vnímanie
  - b) pamäť
  - c) nižšie konvergentné procesy
  - d) vyššie konvergentné procesy
  - e) hodnotiace myslenie
  - f) tvorivé, divergentné myslenie

Klasifikácia úlohy podľa uvedených kritérií je náročná a subjektívna, lebo pri zaradení záleží od mnohých okolností ako sú formulácia úlohy, vedomosti a skúsenosti

žiakov, podmienky vyučovania a organizačný prístup učiteľa (Mindáková, 2008).

### 2.2.3 Preformulovanie úloh

Často sa vyskytujúci nedostatok systému úloh je neúplnosť pestrosti didaktického za-merania cvičení. Ukázalo sa, že nedostatok úloh vo fáze aktualizácie učiva, v prípravnej fáze alebo vo fáze osvojovania učiva sa dá prekonať vytvorením nových úloh vo forme jednoduchých otázok alebo iným zaradením podľa témy, podtémy a elementu učiva. Preformulovaním navyše dosiahneme úpravy kategórií úlohy, či zvýšenie alebo zníženie jej obtiažnosti. Doplnenie chýbajúcich typových úloh sa môže realizovať rozličnými prístupmi, z ktorých vyzdvihujeme tri systematické metódy (Mindáková, 2008):

- a) **Zmena podmienky v zadaní:** najčastejšie mení tému, podtému, element učiva, čím môže mať vplyv na etapu vyučovacieho procesu, kedy sa úloha osvedčí použiť. Konkrétosti dopadu na úlohu závisia od presného textu zadania. Napríklad, zmeniť podtému úlohy z „Príkazy” na „Cyklus” vieme pridaním požiadavky na viacnásobne duplikovanie obrazca vedľa seba. Zmenou formátu vstupu programu, rozdelením viacerých údajov v jednom riadku rozdelením na viac riadkov, sa úloha dostane z podtémy „Refázce” do „Vstupy programu”.
- b) **Tvorba otočenej úlohy:** poskytuje predlohu na prípravu divergentných úloh, ktoré sú spravidla ťažšie na vymyslenie než tie na nižšie myšlienkové operácie. V otočenej úlohe nebudú vyjadrené priamo číselné údaje na dosadenie do vzorca, ale situácia sa ilustruje graficky a žiak musí zvážiť stratégiu riešenia.
- c) **Zmena fabuly úlohy:** sa spolieha pri sprístupnení podstaty úlohy pre iného adresáta na zmenu príbehu slovnej úlohy a zasadenie javov do iného kontextu. Nemení umiestnenie úlohy v zbierke. Výpočty o rozmeroch valcových predmetov môžu takto nadobudnúť dejovú líniu o bareloch nafty, kmeňoch stromov, stenách rotúnd, alebo elektrickom odpore drôtov.

Vyváženie počtu úloh medzi témami a časťami zbierky sa najlepšie dosiahne zmenou podmienky v zadaní. Otvorené úlohy nemajú hojné zastúpenie, pretože zvyknú byť časovo náročné a pre priemerných žiakov za dogmatického spôsobu výučby náročné. Najmä vtedy sa uplatní úprava na obrátenú úlohu. Sady didaktických testov alebo personalizované interaktívne elektronické učebnice, do ktorých je potrebné generovať podobne náročné úlohy z rovnakej oblasti, hojne zužitkujú zmenu fabuly úlohy.

## 2.3 Vzdelávacie štandardy v informatike

Výchovno-vzdelávací štandard sú kritéria vzdelávacej inštitúcie na požadovanú úroveň žiakovho výkonu po kognitívnej, formatívnej a konatívnej stránke. Ciele vzdelávania sú predpísané v štátnom vzdelávacom programe (*ŠVP*), z ktorého školy vychádzajú v školskom vzdelávacom programe (*ŠkVP*). Tvorí ho obsahový štandard vymedzujúci čo sa má žiak naučiť a výkonový štandard s minimálnou normou činnosti žiaka.

V informatike je ŠVP rozdelený na 5 okruhov: algoritmické riešenie problémov, reprezentácie a nástroje, softvér a hardvér, komunikácia a spolupráca, informačná spoločnosť (Minedu, 2023). Konanie internej formy maturitnej skúšky z informatiky nasleduje metodický pokyn cieľových požiadaviek. Dosiaľ sme analyzovali *akým spôsobom* majú učebnice a zbierky úloh predkladať učebnú látku. V súlade so vzdelávacími štandardmi určíme *čo majú obsahovať*.

### 2.3.1 Algoritmické riešenie problémov

Algoritmizácia a programovanie reprezentuje až 70% váhy výslednej známky maturitnej skúšky a najväčší tematický okruh v ŠVP informatiky pre stredné školy, ktorý dostáva v rámcových učebných osnovách ŠkVP najväčší podiel, až približne tretinu z celkovej časovej dotácie (Minedu, 2019). Nemusíme sa pozeráť len na formálne dokumenty, aby sme si uvedomili, že programovanie sa stáva v dnešnom technologickom svete a informačnej revolúcii nepostrádateľnou digitálnou kompetenciou pre život.

ŠVP vyčleňuje 8 tematických celkov algoritmizácie (Minedu, 2023):

- **Analýza problému:** naplánovanie algoritmického riešenia problému rozdelením na menšie časti a opísanie idey v prirodzenom jazyku. Identifikovanie vstupných informácií, očakávaných výstupov a akcií.
- **Jazyk na zápis riešenia:** používanie konštrukcií programovacieho jazyka, vytváranie a interpretovanie zápisov podľa pravidiel syntaxe.
- **Postupnosť príkazov:** skladanie príkazov do poradia na riešenie problému.
- **Nástroje na interakciu:** načítanie neznámej hodnoty na vstupe a zobrazenie výstupu. Ošetrenie prípustného rozsahu alebo formátu hodnôt.
- **Premenné:** priradenie do pomenovanej premennej a jej použitie v aritmetike.
- **Cykly:** odhalenie repetitívnych vzorov so známym a neznámym počtom opako-

vaní. Akumulovanie čiastkových výsledkov v tele cyklu a kombinovanie cyklov s vetvením.

- **Vetvenie:** stanovenie logickej platnosti vlastnej podmienky obsahujúcej boolovskej operácie.
- **Interpretácia zápisu riešenia:** odladenie programu krokováním a opravenie chýb v existujúcich programoch.

Výstižne sa základné pojmy z povinných tematických celkov programovania dajú zhrnúť podľa štruktúr vývojového diagramu na **sekvenciu príkazov, vstupy, výstupy, podmienky a cykly**. V cieľových požiadavkách na maturitnú skúšku sú témy ešte rozšírené o vnorené príkazy cyklu a vetvenia, **podprogramy** s parametrami, lokálnymi premennými, návratovou hodnotou a nerekurzívnym volaním. Navyše sa pridávajú **jednorozmerné polia, textové súbory, zložené údajové štruktúry** a použitie **generátora náhodných čísel** (Minedu, 2019).

### 2.3.2 Existujúce učebnice a zbierky úloh

V prehľade edukačných publikácií, vrátane elektronických, sa upriamime na porovnanie usporiadania uvedenia jednotlivých pojmov, typické formulácie úloh a grafickú úpravu textu. Už učebnica z matematiky pre 3. ročník stredných škôl (Šedivý et al., 1986) a nadväzujúca zbierka úloh (Bušek et al., 1987) z roku 1987 rozoberajú tému algoritmov približne v šírke danej dnešným ŠVP informatiky s okrajovým doplnením o programovanie v jazyku Basic.

Kapitole algoritmy sa venuje 36 strán (z 344 celkovo), kde sú koncepty predstavené v poradí: premenné, podmienené príkazy, príkazy cyklu. Overovanie správnosti algoritmov aplikujú na vývojových diagramoch. Slovné úlohy si zachovávajú ráz príznačne matematických svojim znením aj výpočtovým zameraním. Pokyny sú v rozkazovacom spôsobe 2. osoby množného čísla, ale namiesto ustáleného výrazu „vypočítajte príklad” sa vyskytuje „zostavte algoritmus”. Objavujú sa tu „evergreeny” na poli programovacích cvičení, napríklad určenie najväčšieho čísla na vstupe spomedzi troch, premena jednotiek časových úsekov, nájdenie najväčšieho spoločného deliteľa alebo vypísanie členov rekurentnej postupnosti vyjadrenej vzorcom. Útržky zo zbierky úloh (Obr. 2.1) ukazujú bežný spôsob číslovania úloh.

Odlišný prístup ku grafickej úprave majú knihy zo série „Skúsiš to s ...”, v rámci

**9.32** Daná je určitá suma v Kčs. Zostavte algoritmus, ktorý vypíše spôsob vyplatenia tejto sumy najmenším počtom bankoviek a mincí.

**9.33** Zostavte algoritmus, ktorým premeníte údaj v sekundách na hodiny, minúty a sekundy.

**9.34** Dané sú tri prirodzené čísla. Zostavte algoritmus, ktorým zistíte, koľko je medzi nimi párnych čísel.

**9.35** Dané sú tri prirodzené čísla. Zostavte algoritmus, ktorým zistíte, koľko je medzi nimi nepárnych čísel.

**9.36** Dané sú tri prirodzené čísla. Zostavte algoritmus, ktorým zistíte, či najväčšie číslo z nich je párne.

**9.48** V jednej stanici sa križujú dve linky metra. Prvá linka má interval 2 min 40 s, druhá má interval 3 min 50 s. O 10.00 h prišli obidve linky na stanicu súčasne. Zostavte algoritmus, ktorým určíte všetky časy súčasného príchodu liniek na stanicu v čase od 10.00 h do 18.00 h.

**9.49** Úlohu 9.48 zovšeobecnite pre ľubovoľný interval liniek a pre ľubovoľný časový interval, v ktorom sa určujú časy súčasného príchodu liniek.

**9.50** Jedny hodiny meškajú 5 minút denne, druhé idú denne 3 minúty dopredu. Obidvoje hodín nastavili na správny čas a uviedli do chodu 1. januára 1986 o 00.00 h. Zostavte algoritmus, ktorým zistíte, o koľko dní budú opäť ukazovať rovnaký čas.

(a) Úlohy na podmienený výraz

(b) Úlohy na príkaz cyklu

Obr. 2.1: Ukážky z kapitoly algoritmy v zbierke úloh z matematiky pre 3. ročník SŠ

ktorej boli uvedené knihy programovania pre mikropočítače v Basicu a strojovom kóde (Tatchellová et al., 1990, Wattsová et al., 1991). Cieľové miesta pôsobenia knihy neboli zrejme školy, ale skôr voľnočasové aktivity ako sú počítačové krúžky. Tieto dve knihy sa nápadite odlišujú pestrofarebnými ilustráciami až takmer komiksovým podtónom, kde sú hlavnými hrdinami roboti v ľudskom a hmyzom stvárnení a mimozemšťania. Krátke odseky výkladového textu sú obohatené o motivovanie každého príkazu jednoduchým príkladom priamo pobádajúcim na odskúšanie (Obr. 2.2a). Funkčné bloky kódu rozsiahlejších programov sú priebežne vysvetľované textom so svorkami (Obr. 2.2b), čo môže slúžiť ako dobrý model na prezentovanie riešení v zbierke.

**Premenné**  
Premenné je zariadený úsek pamäte počítača, kam sa ukladá informácia. Pomocou slov LET a INPUT pridáme počítaču, aby do premennej ukladal hodnoty čísel, textu, ako je to dolu na obrázkoch. Nezabudni, že údaje, ktoré zadáš, musia byť správne, inak sa program nebude správať, ako si ty myslíš. Keď povieš, že premenná obsahuje číslo, musíš ho dať do úvodzoviek.

**Výber mena premennej**  
Kde a koľko mená má na výber počítač? Mená premennej musia byť: 1. Krátke 2. Slovné 3. Nie obsahovať medzery 4. Nie obsahovať špeciálne znaky 5. Nie začínať číslom 6. Nie byť rovnaké ako názov programu alebo súboru.

**Slova a premenne**  
Ak povieš PRINT, počítač musí vedieť, čo má vypísať. Preto musíš povedať, čo má vypísať. Ak povieš LET, počítač musí vedieť, čo má uložiť. Preto musíš povedať, čo má uložiť.

**Kozmické bane**  
Vedieš skupinu ľudí, ktorí ťažia rudu na planéte Astron. Rozhoduješ o predaji rudy, kúpe potravín aj o predaji a nákupe bani. Dokážeš riadiť prácu tak, aby boli všetci spokojní, alebo tvoje rozhodnutia povedú ku katastrofe?

**Ako program pracuje**  
Zvolí počet ľudí (L), počet ľudí (P), množstvo poklad (M) a veľkosť zábrzy (Z).  
Vynajde množstvo rudy na sklade.  
Koeficient spokojnosti nastavi na 1.  
Počet rokov nastavi na 1.  
Zvolí cenu bani.  
Zvolí cenu rudy.

(a) Výkladový text o premenných s hlavolamami

(b) Opis kódu programu počítačovej hry

Obr. 2.2: Bohato ilustrovaná kniha o programovaní v jazyku Basic

Učebné texty na webe pod názvom: „Algoritmy a programovanie v Pascale: nielen pre maturantov z predmetu informatika”, tvoria obsiahly prierez prvkov programovacieho jazyka, konkrétne: výraz s premennou, údajové typy, vetvenie, cyklus, cyklus v cykle, procedúry, funkcie, rekurzia, jednorozmerné polia, textový súbor, vyhľadávanie

a triedenie polí, reťazce znakov (Hedvigová, 2007). Učebnica je prehľadne štruktúrovaná. Z obsahu sa hypertextom smeruje na kapitoly, kde je každý nový pojem typograficky zvýraznený podčiarknutím, príkazy jazyka sú odlíšené neproporcionálnym rezom a farbou písma. Po jadre kapitoly nasledujú obvykle 2 vzorové príklady s riešeniami a spravidla 3 priebežné programátorské úlohy (Obr. 2.3), otázky na opakovanie teórie, a úlohy na precvičovanie celej témy. Na konci učebnice je umiestnených 51 jednoduchších úloh na opakovanie a 30 úloh pre náročnejších, ktorým však chýba zmysluplná organizácia náročnosti.

**Príklad 2:**  
Zostavte program, ktorý načíta celé číslo a vypíše či je párne alebo nepárne.

**Riešenie:**

```
uses Crt;
var x:integer;
begin
  write('Zadaj cele cislo: ');
  readln(x);
  if x mod 2 = 0 then write('Zadane cislo je parne')
    else write('Zadane cislo je neparne');
  readln;
end.
```

**Úlohy:**

1. Zostavte program, ktorý načíta dve celé čísla a vypíše menšie z nich.
2. Zostavte program, ktorý načíta celé číslo a vypíše, či to môže byť teplota ľudského tela žijúceho človeka alebo nie.
3. Zostavte program, ktorý načíta strany trojuholníka a zistí a vypíše, či trojuholník s týmito stranami existuje alebo nie.

Obr. 2.3: Riešený príklad z vetvenia nasledovaný úlohami na samostatnú prácu

Slovné úlohy objasňujúce príbehom problémové situácie sú prítomné v súťažiach ako sú Olympiáda v Informatike, organizovaná Národným inštitútom vzdelávania a mládeže (NIVAM), alebo Zenit, Korešpondenčný seminár (KSP) a Letné školy, organizované občianskym združením Trojsten. Vzorové riešenia zadaní vychádzajú v príručkách po skončení kôl. Keďže úlohy bývajú nad rámec základného učiva, tak na vysvetlenie často sa vyskytujúcich algoritmov vznikla *Kuchárka KSP* (KSP, 2022). Ukážka na Obr. 2.4 ilustruje predlohu pre zadanie z KSP, ktoré sa vyznačuje okrem popisu situácie cez krátky dej aj jasným predpísaním vstupov a výstupov.

Uvedenie programovacieho jazyka *Python* do výučby informatiky na stredných školách znamenal dopyt po nových učebných materiáloch, ktoré preložia zápisy hlavne z dosiaľ používaného jazyku Pascal. Medzi učebnicami Pythonu prevláda trend predstavovať programovanie cez procedurálne kreslenie a moduly *tkinter*, *turtle*, niekedy *pygame*. V grafickom programovaní sa pojem cyklu predstavuje oveľa skôr ako podmienky, presne naopak než pri textovom móde.

**Sušenie oblečenia**

Práve ste sa vrátili z výletu s množstvom špinavého oblečenia. Hneď ste ho všetko nahádzali do pračky a oprali. Teraz však prichádza problém. O chvíľu odchádzate na ďalší výlet a musíte teda čo najrýchlejšie usušiť všetky mokré veci.

Sušenie funguje nasledovne. Každé veci sa dá priradiť jej aktuálna vlhkosť  $v_i$ . Ak bude vlhkosť nulová, vec sa pokladá za vysušenú a ďalej neschne (lebo nemá z čoho). Ak je nejaká vlhká vec na vzduchu, každú minútu stratí jednu vlhkosť. Našťastie však máte jeden radiátor, na ktorý sa zmestí práve jeden kus oblečenia. Ak je vec položená na radiátore, každú minútu stratí  $k$  vlhkosti. Ak by v nejakom momente mala vlhkosť menšiu ako  $k$ , tak uschne úplne.

Veci na radiátore viete každú minútu vymeniť a nezožerie vám to žiaden čas. Zistite, ako najrýchlejšie viete usušiť všetky mokré veci, ak budete používať radiátor optimálne.

**Úloha**

Na vstupe dostanete popis vlhkosti všetkých vecí a parameter  $k$  radiátora. Zistite najmenší čas potrebný na úplné usušenie všetkých vecí.

**Vstup**

Na prvom riadku je číslo  $n$  ( $1 \leq n \leq 10^5$ ) – počet mokrých vecí.  
 Na druhom riadku je  $n$  celých čísel  $v_i$  ( $1 \leq v_i \leq 10^9$ ) – vlhkosti jednotlivých vecí.  
 Posledný riadok obsahuje číslo  $k$  ( $1 \leq k \leq 10^9$ ) – množstvo vlhkosti, ktoré vec stratí počas jednej minúty na radiátore.

**Výstup**

Jedno celé číslo – najmenší počet minút, za ktoré vieme usušiť všetky kusy oblečenia.



**Upozornenie:** Dávajte si pozor, na úspešné riešenie je treba správne použiť 64 bitovú premennú – long.

**Priklady**

vstup	výstup
3 2 3 9 5	3 <i>Veci s vlhkosťou 2 a 3 necháme uschnúť normálne, vec s vlhkosťou 9 dáme na radiátor. Všimnite si, že za 2 minúty všetko usušiť neviem.</i>

Obr. 2.4: Úloha letnej školy KSP na binárne vyhľadávanie s úvodným príbehom

Kučera a Výboštok dali dohromady trojdielnu sériu učebníc „Programujeme v Pythone“, v slovenskom a anglickom jazyku so zodpovedajúcimi príručkami pre učiteľov a testami k učebnici. Vypracovali aj zbierku 64 riešených úloh k maturite z informatiky „Maturujeme v Pythone“. Vychádzali z potrieb aktívnych učiteľov z Klubu učiteľov informatiky. Osnova prvého diela učebnice začína grafickými príkazmi (Obr. 2.5a) a ďalej sa skladá z premenných, opakovaní častí programu, podprogramov, klikania myšou a ovládania klávesnicou, podmienených príkazov, časovača. Na záver učebnice je polročné snaženie žiakov završené tvorbou jednoduchých hier (Kučera, 2016).

6. úloha	Riešenie
<p>16 Pre obe elipsy nastavte rôzne farby výplne, obrysov a tiež hrúbky čiar.</p> <p>17 Nakreslite vlajku Japonska.</p> <p>18 Nakreslite snehuliaka.</p> <p>19 Nakreslite symetrickú tvár s očami, nosom a ústami:</p> <div style="text-align: center;">  </div> <p>20 Doplňte k tvári okuliare.</p>	<pre>from turtle import * luc = 50 telo = 35 pencolor('orange') pensize(5)  for i in range(9):     fd(luc)     bk(luc)     rt(360/9)     pensize(telo)     pencolor('yellow')     fd(0)</pre> <div style="text-align: center;">  </div>

(a) Kreslenie obdĺžnikov

(b) Cyklus a korytnačia grafika

Obr. 2.5: Úlohy na programovanie grafiky v jazyku Python

Blaho a Salanci pripravili pracovné listy *abcPython* na 20 vyučovacích hodín. Pri ich preberaní nepočítajú s výkladom učiteľa. Dostupné sú aj metodické materiály ku



listom. Postupne sa objavujú témy, kde sa prelína textový a grafický režim: interaktívne zadávanie príkazov, výrazy, premenné, výpisy, kreslenie, náhoda, výrazy v cykle, elipsy, vetvenie, podprogramy, kreslenie myšou (Blaho et al., 2019a, Blaho et al., 2019b).

Mészárosová vytvorila metodickú príručku pre vyučovanie základov programovania, kde cez Python rozvíja na rozpätí 16 vyučovacích hodín korytnačiu grafiku (Obr. 2.5b). Využíva tým oboznámenosť žiakov s korytnačkami v jazyku Logo z druhého stupňa základnej školy. Rovnako začína predstavením grafickými pokynov na pohyb a kreslenie korytnačkou. Nasledujú premenné, for cyklus, funkcie, funkcie s parametrami, poloha korytnačky, náhodná poloha a vetvenie. Na opakovanie osvojených zručností slúži projekt kreslenia pohľadnice (Mészárosová, 2017).

### 3 Cieľ a metodika práce

Cieľom záverečnej práce je zostaviť rozšíriteľnú **zbierku úloh z programovania** so vzorovými riešeniami v predmete informatika pre stredné školy. Primárny zámer súčasne vyžaduje uspokojenie nasledujúcich požiadaviek na navrhnutý systém úloh pre pedagogickú prax:

- pokrytie základného učiva v súlade so ŠVP a cieľovými požiadavkami
- poskytnutie metodiky na spôsob zaradenia ďalších úloh do nášho systému úloh
- čitateľnosť textu zadania primeranej žiakom vo veku 15 - 18 rokov
- príbehovosť opisu problémovej situácie v znení zadania
- vzorové riešenia v programovacom jazyku Python
- zobrazenie na webovej stránke s efektívnym využitím hypertextu

Splnenie vytýčených cieľov sa opiera o rešerš informačných zdrojov vyhľadaných prostredníctvom plnotextových vyhľadávačov webu a knižníc, na základe rád školiteľky a osobného povedomia o oblasti. Knihy a články boli podrobené procesu analýzy, kedy sa vyselektovali relevantné definície, charakteristiky, kategorizácie a postupy. Metóde porovnania sa podrobili existujúce učebné texty základov programovania.

Vlastná tvorba zbierky úloh prebiehala syntézou psychologických východísk porozumenia textu, známych typových úloh a známej metodiky na preformulovanie úloh. Hodnotenie kvality textu úloh v zbierke sa uskutočnilo kvantitatívne i kvalitatívne. Úroveň čitateľnosti sa indikatívne vyčíslila pomocou Gunning fog indexu (Vzorec 2.1). Kvalitatívne sa pozorovali problémy žiakov pri riešení úloh v zbierke na vyučovacej hodine plynúcej z neporozumenia formulácie zadania. Metodika klasifikácie úlohy v systéme vychádza z kritérií témy, podtémy, elementu, didaktickej funkcie a kognitívnej úrovne podľa Mindákovovej (Sekcia 2.2.2).



## 4 Výsledky práce

Medzi najdôležitejšie tu nachádzajúce sa dielo patria problémové úlohy z programovania pre začiatočníkov na stredných školách. V súvislosti s našou zbierkou úloh rozoberieme adoptovanú anatóniu úlohy a dôsledky vyplývajúce z terajšieho zoradenia cvičení. Vychádzame pritom z teoretických základov tvorby učebníc a požiadaviek na pestrosť aplikačného prostredia pre pojmy za účelom pritiahnutia záujmu učiaceho sa subjektu.

### 4.1 Zbierka úloh

Zhromaždený súbor zadaní z programovania sa skladá počtom z 54 úloh s riešeniami, ktoré sú rozčlenené do tematických častí nerovnomerne. Oddiel premenné má 11 úloh, podmienky - 9 úloh, cykly - 9 úloh, náhodné čísla - 3 úlohy, reťazce a zoznamy - 9 úloh, súbory - 5 úloh, funkcie - 8 úloh. Usporiadanie celkov pochádza čiastočne zo vzdelávacích štandardov ale aj zo zvyklostí.

Nepravidelnosti v kvantite častí nastali kvôli organickému a postupnému pribúdaniu úloh, ktoré reagovalo na potreby žiakov vedúcich ku zvládnutiu predmetu. Na kostru úloh sa dodatočne nabalili aj niektoré testové problémy, cvičenia na doma a upravené učivo z iných predmetov, ktoré trápilo žiakov pred písomkami. K zdanlivo relatívne menšiemu objemu časti súbory poznamenáme, že v skutočnosti prvá úloha spočíva v prispôbení všetkých úloh z predošlej témy na súbory. Na súbory je preto k dispozícii až 13 úloh. Ďalej uvážame znenia zadaní úloh.

#### 4.1.1 Premenné

**Premenná** je taká krabička na odkladanie čísel alebo slov, ktoré si potrebujeme zapamätať na dokončenie činnosti. Premenné sa líšia svojím dátovým typom. Premenná dostane svoj typ cez priradenie, čiže vtedy keď prvýkrát do nej niečo uložíme. Typ hovorí o tom, čo sa vo vnútri premennej nachádza.

Základné typy premenných sú:

- **Logická hodnota** (*bool*) - môže mať len dve hodnoty - pravda (*True*) alebo nepravda (*False*)
- **Celé číslo** (*int*) - ukladáme sem ľubovoľné kladné a záporné celé čísla (97)
- **Desatinné číslo** (*float*) - Líšia sa od celých čísel spôsobom uloženia (3.14159)
- **Reťazec** (*str*) - Označujeme ich úvodzovkami alebo apostrofmi a väčšinou predstavujú text napísaný na klávesnici alebo zobrazený na obrazovke (*“Učím sa programovať!”*).

## 1. Pozdrav

Skladáš si stavebnicu robotického domáceho miláčika, ktorý je takmer dokonalý. Má telo, končatiny, hlavu a vie kráčať po stole. Keby však sa naučil aj hovoriť, to by bol potom poriadny spoločník. Každý dobrý rozhovor sa začína pozdravom. Napíš program, ktorý ťa pozdraví po napísaní mena na klávesnici. Doplň tiež, aby sa program s tebou aj rozlúčil.

Vstup:

Ako sa voláš?:

Výstup:

Ahoj

## 2. Básnik

Rozkriklo sa, že píšeš pekné básničky na rôzne príležitosti. Prichádza ti čím ďalej viac prosieb od kamarátov a známych, či im nevytvoríš peknú rýmovačku. Vymýšľať kreatívne texty je niekedy veľká námaha, tak ti napadne, že stačí meniť len rým. Napíš program, ktorý za teba slovo vloží do predlohy básne.

Vstup:

Napíš slovo, ktoré sa rýmuje so slovom strach:

Výstup:

Tu je báseň:  
Z počítačov mával som vždy strach,  
teraz som však šťastný ako .

### 3. Pozvánka

Budúci víkend organizuješ veľkolepú narodeninovú párty a rozposielaš na ňu pozvánky. Okrem mena hosta potrebuješ meniť aj čas konania oslavy. Máš totiž skúsenosti, že nie všetci chodia načas. Každý pozvaný si má priniesť okrem darčeka aj jednu špeciálnu vec. Napíš program, ktorý doplní takúto pozvánku na mieru.

**Vstup:**

Meno kamaráta:   
 Čas oslavy:   
 Prinesie okrem darčeka:

**Výstup:**

Ahoj ,  
 pozývam ťa na moju narodeninovú párty.  
 Bude sa konať 12.4. o .  
 Nezabudni priniesť  a pekný darček.  
 Teším sa na teba! :)

### 4. Teplota vo Farenheitoch

Prišiel si na dovolenku do Spojených štátov amerických. Chystáš sa na krátky výlet von, ale nevieš ako sa máš obliecť. Na teplomeroch sú napísané len stupne Fahrenheita. Napíš program, ktorý ich premení na stupne Celzia s presnosťou na dve desatinné miesta.

**Vstup:**

Vonku je °F:

**Výstup:**

Doma by bolo na teplomeri  °C.

### 5. Hlboká roklina

Stojíš nad hlbokým údolím za zábradlím a uvažuješ ako odmerať jeho hĺbku. Vtom si spomenieš na svoje vedomosti z fyziky. Zoberieš si do ruky váľajúci sa kameň a pustíš ho priamo do rokliny. Zároveň stlačíš stopky, ktorými zmeriaš čas do dopadu v sekundách. Kameň padá nadol voľným pádom. Stopky zastavíš pri započutí rachotu z nárazu. Pri výpočte zanedbáme rýchlosť zvuku, ktorou sa rachot rozšíri až k nám.

**Vstup:**

Čas do dopadu kameňa:

**Výstup:**Hĺbka rokliny je  metrov.

## 6. Vedro s vodou

V rodinnom dome ste ekologicky uvedomelí, lebo zachytávate dažďovú vodu z odkvapů na polievanie záhrady. Minulú noc vám napršalo do nádrže veľa vody. Keď bude o pár dní suchšie mama ťa pošle poliať rastliny uzavretým vedrom valcového tvaru. To naplníš vždy až po okraj. Zaujímá ťa, aký objem naberieš na jedno naplnenie. Rozmery valcového vedra vieš odmerať pravítkom. Napíš program, ktorý zráta koľko sa zmestí vody do rôzne veľkých vedier.

**Vstup:**Výška vedra (cm):   
Priemer dna (cm): **Výstup:**Do vedra sa zmestí  litrov vody.

## 7. Cesta autom

Tešíš sa na očakávaný výlet autom po Európe. Pri plánovaní trasy chceš zistiť akou rýchlosťou musíte priemerne cestovať, aby ste od rána stihli navštíviť všetky miesta. Večer však musíte prísť včas do hotela, aby vás ubytovali. Napíš program, ktorý ti s tým pomôže.

**Vstup:**Dĺžka cesty (km):   
Odchod z domu (hodina):   
Príchod do hotela (hodina): **Výstup:**Auto pôjde priemernou rýchlosťou  km/h.

## 8. Kúpalisko

Začína sa horúca letná sezóna. Prevádzka kúpaliska musí pred otvorením napustiť bazény. Všetky bazény v areáli sú kvádrového tvaru, ktorých rozmery poznáme. Vedúceho kúpaliska zaujíma spotrebovaná voda pre bazén, keď bude napustený pod okraj. Voda nie je zadarmo, preto si chcú pripraviť dosť peňazí, aby za ňu zaplatili.

**Vstup:**

Dĺžka bazéna (m):   
 Šírka bazéna (m):   
 Hĺbka bazéna (m):   
 Hĺbka hladiny pod okrajom (cm):   
 Cena za m<sup>3</sup> vody v eurách:

**Výstup:**

Na bazén sa minie  litrov vody.  
 Voda bude stáť  eur.

## 9. Maľovanie

S rodičmi sa sťahuješ do nového bytu. Dali ti za úlohu kúpiť si farbu na vymaľovanie izby. Nástroj na rýchle počítanie množstva farby by sa hodil asi aj profesionálnym maliarom. Vytvor program na vypočítanie plochy stien a stropu bez okna a podlahy.

**Vstup:**

Rozmery miestnosti  
 Dĺžka (cm):   
 Šírka (cm):   
 Výška (cm):   
 Rozmery okna  
 Šírka (cm):   
 Výška (cm):   
 Výdatnosť farby (m<sup>2</sup>/kg):

**Výstup:**

Maľovať budeš plochu  m<sup>2</sup>.  
 Kúp  kg farby.

## 10. Chemikálie

Chemici v laboratóriu bežne zmiešavajú roztoky, aby dosiahli správny pomer želanej látky. Roztoky sú opísané svojou hmotnosťou ( $m$ ) a hmotnostným zlomkom rozpustenej látky v rozpúšťadle ( $w$ ). Viaceré látky odlíšime dolným indexom ( $m_1$ ). Hmotnosť sa uvádza v gramoch a hmotnostný zlomok v percentách. Napíš program na opísanie vlastností výsledného roztoku. Na výpočet použi tieto rovnice:

$$m_3 = m_1 + m_2$$

$$m_3 \cdot w_3 = m_1 \cdot w_1 + m_2 \cdot w_2$$



**Vstup:**

Hmotnosť roztoku č.1 (m1)?   
Hmotnostný zlomok roztoku č.1 (w1)?   
Hmotnosť roztoku č.2 (m2)?   
Hmotnostný zlomok roztoku č.2 (w2)?

**Výstup:**

Výsledný roztok má hmotnosť  g.  
Hmotnostný zlomok rozpustenej látky je  %.

## 11. Brzdzenie

V poslednej dobe sa objavuje na trati viac nebezpečných zrážok. Rušňovodiči ťa požiadali, aby si zistil ako rýchlo a ďaleko pred prekážkou dokáže vlak zastaviť. Vlaková súprava ide pred brzdením svojou stálou rýchlosťou v kilometroch za hodinu. Hmotnosť vlaku tvorí súčet hmotností lokomotívy a všetkých vagónov. Brzdy na kolesách majú spoločnú brzdnú silu uvedenú v Newtonoch na tonu. V programe využiješ nasledovné fyzikálne vzťahy:

- Kinetická energia pohybujúceho sa vlaku (práca potrebná na zabrzdenie):

$$W = E_k = \frac{1}{2} \cdot m \cdot v^2$$

- Brzdná dráha pri brzdnjej sile  $F_b$ :  $s = \frac{W}{F_b \cdot m}$

- Čas na zastavenie vlaku pri rovnomernom spomalenom pohybe:  $t = \sqrt{\frac{2 \cdot s}{F/m}}$

**Vstup:**

Vlaková súprava  
- Rýchlosť (km/h):   
- Hmotnosť lokomotívy (t):   
- Hmotnosť vagóna (t):   
- Počet vagónov:   
- Brzdná sila (N/t):

**Výstup:**

Vlaková súprava má hmotnosť  ton.  
V rýchlosti  km/h zabrzdí na vzdialenosť   
metrov.  
Brzdzenie bude trvať  sekúnd.

### 4.1.2 Podmienky

Podmienky sú ako križovatky na ceste. Podľa toho kam chceme ísť, sa rozhodneme, ktorou cestou pôjdeme ďalej. Aby sme sa uistili, že máme ten správny smer (vetva podmienky) pýtame sa vždy logickú otázku. Otázka používa údaje uložené v premenných.

#### 1. Heslo

Tvoj dom na strome už vykradlo pár nezvaných návštevníkov. Vymyslel si preto spôsob ako povoliť návštevu len overeným osobám. Tie musia poznať tajné heslo. Napíš program, ktorý slovne privíta členov a odoženie zlodejov.

**Vstup:**

Stoj! Povedz Heslo!

?

**Výstup:**

Vstúp, priateľ

(alebo Zmizni kade ľahšie)

#### 2. Najväčšie číslo

Na lúke sa hrajú šípky. Hráči si zapisujú dosiahnuté skóre na tabuľu. Dnes proti sebe hrali v partii traja protihráči. Napíš program, ktorý označí hráča s najväčším získaným počtom bodov.

**Vstup:**

1. skóre:

2. skóre:

3. skóre:

**Výstup:**

Najväčšie skóre  bodov má  hráč.

#### 3. Vhodné oblečenie

Módni poradcovia vyšli z módy a ich prácu prebrali počítače. Na základe počasia a príležitosti odporúčajú vhodný outfit. Vymysli pár tipov pre rôzne situácie a začni radiť.

**Vstup:**

Ako je vonku?:

Kam ideš?:

**Výstup:**

Určite si nezabudni  a tiež si vezmi .

### 4. Morský vánok

Kapitán plachetnice na otvorenom oceáne musí mať vždy prehľad odkiaľ fúka vietor, aby odkormidloval do vytúženého cieľa. Príliš silné závaný vetra môžu byť nebezpečné pre posádku. Polámať lodné sťažne, potrhať plachty, či zaplaviť palubu. Cez rádio dostáva plavidlo každý deň správy o predpovedi sily vetra v Beafortovej stupnici. Sila vetra je ňou vyjadrená do dvanástich stupňov od bezvetria až po orkán. Napíš program, ktorý kapitánovi vysvetlí stupeň vetra. Podľa stupnice určíme jeho pomenovanie, rýchlosti v námorných uzloch a očakávanej výšky vln.

**Vstup:**

Sila vetra na Beaufortovej stupnici:

**Výstup:**

Vietor sa nazýva .

Vietor má rýchlosť  kt.

Očakávaná výška vln je  m.

### 5. Pokazený rozpis

Továreň na železnú rudu dostala nový časový rozpis vylepšeného technologického procesu. Spracovanie zvyčajne trvá dlhšie ako hodinu. Nehodí sa im teda mať časy napísané iba v minútach. Rozpiš programom minúty na dni, hodiny, minúty pre jednoduchšie čítanie rozpisu. Vynechaj nepotrebné časové údaje.

**Vstup:**

Trvanie (min.):

**Výstup:**

=  d.  hod.  min.

### 6. Hovoriaca kalkulačka

Výpočty neboli nikdy väčšia zábava. Teda aspoň s kalkulačkou, ktorá namiesto čudných matematických čmáraníc hovorí ľudskou rečou. Vytvor program pre kalkulačku, ktorá si vypýta dve čísla. Tie bude ich vedieť sčítat alebo odčítat podľa slovného pokynu.

**Vstup:**

Som hovorica kalkulačka a rada počítam!  
 Povedz mi prvé číslo:   
 Potrebujem ďalšie číslo:   
 Chceš ich sčítať alebo odčítať:  (sčítať alebo  
 odčítať)

**Výstup:**

Výsledok tvojho príkladu:  (plus alebo mínus)  
 je .

## 7. Chaos v lístkoch

Vyznať sa v linkách mestskej hromadnej dopravy si vyžaduje dlhoročné skúsenosti. Treba oplývať aj riadnou dávkou trpezlivosti. Ľahko sa nám stane, že omylom nasadneme do autobusu a hneď sa vydáme na okružnú jazdu po siedmich divoch sídliska. Horší zážitok je stretnutie revízora po zistení, že máme nesprávny lístok alebo že nemáme žiaden ... Postávaš pri automate na lístky a nevieš sa vysomáriť z množstva časov a zón v ponuke. Napíš program, ktorý podľa počtu zónu a trvania ceny vypíše cenu zľavneného lístka. Nájdi na internete aktuálnu tarifu MHD v tvojom meste.

**Vstup:**

Popíš mi svoju cestu s MHD  
 Koľko zón prejdeš?:   
 Koľko minút má trvať cesta?:

**Výstup:**

Zľavnený lístok stojí  eur.

## 8. Kvadratická rovnica

Matematika v škole dokáže byť poriadna otrava. Hlavne, keď od rána do večera nič iné nerobíš ako počítáš príklady na kvadratické rovnice. „Načo mám ten počítač“, pomyslíš si večer vo svetle stolnej lampy. Pre zadané koeficienty  $a$ ,  $b$ ,  $c$  predpisu  $ax^2 + bx + c = 0$  napíš program, ktorý vypočíta jej korene a vrchol paraboly.

**Vstup:**

Koeficienty kvadratickej rovnice:  
 $a =$    
 $b =$    
 $c =$

**Výstup:**

$$\square x^2 + \square x + \square = 0$$

$$x1 = \square$$

$$x2 = \square$$

$$v[\square; \square]$$

## 9. Trojuholníky

Trojuholník je mýtická bytosť, o ktorej je vždy treba zistiť. Nesmieme použiť pravítko, lebo to by nás čakala príliš jednoduchá výzva. Veď bez rysovania zistíme o tejto trojcipej paráde všeličo. Hoci aj keď jej chýbajú niektoré rozmery.

- Ak je to možné, doplň chýbajúce informácie pre ľubovoľný trojuholník (zadaný ako SSS) ako sú dĺžky strán a výšok, veľkosti uhlov, obsah a obvod. Využi trojuholníkovú nerovnosť, sínusovú vetu, kosínusovú vetu a vzorec na výpočet obsahu trojuholníkov.
- Rozšír program aj pre ostatné vety o trojuholníkoch: SUS, USU, UUS.

**Vstup:**

Zadajte strany ľubovoľného trojuholníka:

$$a = \square$$

$$b = \square$$

$$c = \square$$

**Výstup:**Strany:  $a = \square$ ;  $b = \square$ ;  $c = \square$ Uhly:  $\alpha = \square^\circ$ ;  $\beta = \square^\circ$ ;  $\gamma = \square^\circ$ Výšky:  $v(a) = \square$ ;  $v(b) = \square$ ;  $v(c) = \square$ 

$$O = \square$$

$$S = \square$$

Trojuholník je:  $\square$ ,  $\square$ 

### 4.1.3 Cykly

Obrovský potenciál počítačov tkvie v bezchybnom neúnavnom vykonávaní presne zadaných inštrukcií. **Cykly** umožňujú opakovať rovnaký postup ľubovoľný počet krát a tým efektívne odstraňovať rutinnú prácu.

### 1. 100-krát napíš

Za prehrešky proti školskému poriadku sa stalo populárnym trestom ručné prepisovanie mravoučnej vety stokrát. Stalo sa to tak neznesiteľné, že si zhotovil robota na pomoc záškodníkom. Chýbajú mu len príkazy, čo má vlastne robiť.

**Vstup:**

Musím napísať:   
Toľkoto krát:

**Výstup:**

...

### 2. Hodnotenie

Filmoví a gastronomickí kritici zavŕšia namáhavý deň udelením číselného skóre k ich recenziám. Pre lepší efekt v časopise potrebujú nakresliť hviezdičky namiesto čísla. Pomôž im programom.

**Vstup:**

Skóre:

**Výstup:**

### 3. Pyramída

Mayská civilizácia sa mohla pýšiť v čase svojho najväčšieho rozmachu všelijakými na tú dobu pokrokovými vymoženosťami. Doteraz sa ospevuje ich písmo, sofistikovaný kalendár a znalosti z astronómie. V mestách stavali mohutné chrámové pyramídy na náboženské obrady. Preniesol si sa späť v čase a ocitol si sa pri plánovaní pyramídy. Stavitelia chcú nakresliť jej plány, aby vedeli ako majú poskladať kamenné bloky. Napíš program, ktorý vypíše hviezdičky do tvaru pyramídy podľa jej výšky.

**Vstup:**

Výška pyramídy:

**Výstup:**

```
  *
 ***
*****
*****
```

#### 4. Smaragd

Nie všetko, čo sa blyští je zlato. Drahokamy ako rýdzy zelený smaragd však ulahodia oku podobne. Hruda horniny sa najprv musí vybrúsiť napríklad do amuletu, ktorý sa môže stať parádou náhrdelníku. Prešibaný zlatník nakupuje pre zákazníkov smaragdové amulety v tvare osemstenu. Ten z boku vyzerá takmer ako kosoštvorec. Zlatník ho chce porovnávať s ideálnym tvarom, aby mohol dohodnúť nižšiu cenu, keď ho bude chcieť dodávateľ podviesť. Napíš program na vykreslenie „smaragdu” z hviezdíčiek podľa zadanej veľkosti.

**Vstup:**

Veľkosť smaragdu:

**Výstup:**

```
  *
 ***
*****
 ***
  *
```

#### 5. Duté vnútro

Staviteľov pyramíd začalo zaujímať zariaďovanie ich vnútra. Do posvätného chrámu sa predsa musia zmestiť všetky bohatstvá, ktorými si budú uctievať božstvá. Program tentokrát vykreslí hviezdíčkovú pyramídu bez výplne.

**Vstup:**

Výška pyramídy:

**Výstup:**

```
  *
 * *
*   *
*****
```

## 6. Mriežka slov

Tapety na stenu sa objavujú v najrozmanitejších podobách od hypnotických špirál cez kvetinové lúky až po hotové umelecké diela. Ešte nikoho nenapadlo si v obývačke natapetovať nekonečný zástup slov. Načítaj v programe šírku tapety a slovo, ktoré sa bude na každom riadku a v stĺpci na nej opakovať.

**Vstup:**

Počet riakov a stĺpcov:   
Opakovať slovo:

**Výstup:**

```
ano ano ano ano
ano ano ano ano
ano ano ano ano
ano ano ano ano
```

## 7. Rám

Moderné umenie má svojich bezbrehých obdivovateľov aj zásadových neznalcov. Krásny obraz môžu tvoriť hoc opakujúce sa slová. Na zosilnenie dojmu by mali byť pekne zarámované. Na prvý a posledný riadok a stĺpec doplní program symboly „#”. Tie poskytnú rám pre zo mriežku slov.

**Vstup:**

Počet riakov a stĺpcov:   
Opakovať slovo:

**Výstup:**

```
### ### ### ###
### ano ano ###
### ano ano ###
### ### ### ###
```

## 8. Malá násobilka

K výbave každého žiaka základnej školy patrí tabuľky malej násobilky. Vytvor takúto tabuľku obsahujúcu každý násobok od 1x1 po 10x10, aby si pomohol všetkým malým počtárom.



**Výstup:**

1	2	3	4	5	6	7	8	9	10
2	4	6	8	10	12	14	16	18	20
3	6	9	12	15	18	21	24	27	30
4	8	12	16	20	24	28	32	36	40
5	10	15	20	25	30	35	40	45	50
6	12	18	24	30	36	42	48	54	60
7	14	21	28	35	42	49	56	63	70
8	16	24	32	40	48	56	64	72	80
9	18	27	36	45	54	63	72	81	90
10	20	30	40	50	60	70	80	90	100

## 9. Sporenie

Na letnej brigáde si zarobil peniaze, ktoré si chceš usporiť. Porovnáš ponuky bánk a hľadáš najvýhodnejší plán. Vytvor si sporiacu kalkulačku, ktorá vypíše vývoj tvojich finančných prostriedkov do budúcnosti. Bude vychádzať z tvojho počiatočného vkladu, ročnej úrokovej sadzby, typu úročenia a peňazí, ktoré chceš mať na konci.

**Vstup:**

Počiatočný vklad v eurách:	<input type="text"/>
Úroková sadzba p.a. v %:	<input type="text"/>
Typ úročenia (jednoduché / zložené):	<input type="text"/>
Žiadaná suma v eurách:	<input type="text"/>

**Výstup:**

Rok	Suma	Úrok
1.	<input type="text"/> Eur	<input type="text"/> Eur
2.	<input type="text"/> Eur	<input type="text"/> Eur

### 4.1.4 Náhodné čísla

Pri tvorbe simulácií sa bez náhodného čísla nezaobídeme. Umožňujú vniesť nečakané javy a rôznorodosť do inak nemienných scén. Nesmierne poslúžia v hrách, kde dovoľujú meniť napríklad výskyt monštier, či pokladov.

#### 1. Hádzanie kockou

Hranie človeče nehnevaj zaberie pokojne celé popoludnie. Chvíľa nepozornosti stačí, aby sa kocka nadobro zatúlala pod ťažký gauč. Vytvor si namiesto zapadnutej kocky

program, ktorý napodobní jej hod. Po stlačení klávesy Enter sa nakreslí kocka s padnutým číslom. Hodené číslo je po každom spustení programu iné.

**Vstup:**

HOD

**Výstup:**

```
+-----+
|  #   #  |
|   #   |
|  #   #  |
+-----+
```

## 2. Háďaj číslo

Háďaj na čo práve myslím bude až do vynálezu telepatie zábavná kratochvíľa. Okrem osobností, vecí a miest sa zvyknú tipovať aj čísla. Nechaj program náhodne vybrať číslo od 0 po 100. Hráč bude ho háďať až pokým neuháďne. Poskytni mu po každom pokuse nápoved', či povedal priveľa alebo primálo. Potom doplň do programu rôzne náročnosti. Môže ísť o napríklad s možnosť nastaviť rozsah čísel alebo maximálny počet tipov.

**Vstup:**

Háďaj číslo:   
 Háďaj číslo:   
 Háďaj číslo:

**Výstup:**

Málo  
 Veľa  
 Výborne. Uháďol si!

## 3. Opakovanie násobilky

Vďaka tvojej tabuľke malej násobilky sa malí školáci mohli naučiť násobiť. Ako dobre to vedia, musíš teraz otestovať. Vygeneruj dve čísla od 1 do 10 do príkladu na násobenie. Over správnosť žiackovej odpovede.

**Vstup:**

Koľko je  x ?  
 =   
 Chceš ďalší príklad (a / n)?

**Výstup:**

Správne - len tak ďalej / Nesprávne - háďaj znovu

### 4.1.5 Reťazce a zoznamy

**Zoznam** (tiež aj **Pole**) je množina údajov zaznamenaných spolu pod jedným menom. Každý údaj poľa sa nazýva **prvok** a poradie jeho pozície sa nazýva **index**. **Reťazce** sa správajú podobne ako zoznamy, ale ich prvkami sú jednotlivé **znaky**.

#### 1. Vymeň písmeno

Niektó ti posiela správy s diakritikou, ale po ceste sa vždy prekrúti jedno písmeno. Texty obsahujú aj pekné básne, ktoré si chceš vytlačiť a pripnúť na nástenku. Pokazený znak však kazí celkový dojem z diela. Zameň zadané chybné písmeno v celom reťazci.

**Vstup:**

Správa:   
Za chybné písmeno:   
Vymeň:

**Výstup:**

Opravené!

#### 2. Cenzúra

Prišla tvrdá cenzúra s nariadením, že nikto už nesmie vidieť žiadnu samohlásku. Nahraď každý priestupok vo vstupnom texte iným špeciálnym znakom.

**Vstup:**

Správa:   
Samohlásku nahraď:

**Výstup:**

Cenzurované:

#### 3. Počítanie slov

Do redakcie miestnych novín chodia dennodenne články, vtipy, poviedky a príbehy zo života od verných čitateľov. Aby mohli byť uverejnené potrebujú sa zmestiť do vyhradeného priestoru. Vypíš počet znakov, slov, viet a normostrán (=1800 znakov), aby sa príhody rýchlejšie rozšírili medzi ľuďmi.

**Vstup:**

Článok:

**Výstup:**Znaky: Slová: Vety: Normostrany: 

#### 4. Najdlhšie slovo

Debatný spolok usporiadal súťaž o nájdenie najdlhšieho slova, ktoré sa kedy vyskytlo v historických prejavoch. Zaujali ťa odmeny, ale nechce sa ti prehrabávať knižnicou starých záznamníkov. Prácu si preto uľahčíš. Nájdi najdlhšie slovo v ľubovoľnom reťazci.

**Vstup:**Rečnícky prejav: **Výstup:**Najdlhšie slovo v ňom: 

#### 5. Výskyt písmen

Dlho do noci čítaš časopisy o umelej inteligencii a fascinuje ťa jej schopnosť rozprávať sa s človekom. Na vytvorenie viet na danú tému potrebuje mať prehľad o percentuálnom výskyte hlások v texte. Spočítaj a vypíš zoznam početnosti písmen v reťazci.

**Vstup:**Článok: **Výstup:**A:  23.2 %B:  11.5 %C:  8.9 %

...

Z:  0.3 %

#### 6. Histogram

Počas predošlého pokusu s početnosťou písmen si všimneš, že každé ďalšie písmeno v zozname sa objavuje oveľa menej než očakávaš. Vykresli hviezdičky namiesto počtu percent. Over si tak svoje pozorovanie graficky.

**Vstup:**Článok:

**Výstup:**

A :

E :

I :

...

X :

## 7. Nákupný košík

Na veľkých nákupoch sa často zide prehľadný zoznam s tým, čo doma treba. Pýtaj si položky s ich cenami až kým sa nerozhodneš, že máš spísané všetko. Zobraz prehľadnú orámovanú tabuľku s údajmi, podobne ako na pokladničnom bločku. Sú to názov tovaru, DPH tovaru, cena tovaru s DPH, peňazí spolu za nákup.

**Vstup:**

Čo kúpiť?:

Cena ?:

**Výstup:**

+-----+-----+-----+			
Tovar	DPH	Cena s DPH	
+-----+-----+-----+			
Chlieb	0,20	0,98	
+-----+-----+-----+			
...	...	...	
+-----+-----+-----+			
CELKOM	0,20	0,98	
+-----+-----+-----+			

## 8. Akronym

SMS-ky rapídne zdraželi. Napadlo ti, že bude lepšie posilať slovné spojenia ako skratky. Zo zadaných slov vytvor akronym, ktorý vznikne ponechaním len začiatočných písmen každého slova.

**Vstup:**

Slovné spojenie:

**Výstup:**

Skratka:

## 9. Veľa opakovania

Roboti rozvážajú pizzu po meste. Popri tom si zapisujú zmenu smeru pre postupné vylepšovanie trás na k častým zákazníkom. Keďže sa firme darí, nachodili roboti toho už riadne veľa. Všetky záznamy o ich cestách sa im ani nezmestia do pamäti. Všimneš si, že si značia každý jeden krok, čiže sa často opakujú. Nahraď postupnosť za sebou idúceho písmena, písmenom a jeho počtom výskytu.

**Vstup:**

Cesta robota: NNNNNNSSSSSSSSSSWWWNNN

**Výstup:**

Skomprimované: 6N11S4W3N

### 4.1.6 Súbory

**Súbor** je zoskupením súvisiacich údajov, ktoré sú uložené na disku počítača. Oproti načítaniu vstupu z klávesnice majú výhodu hlavne pri spracovaní a uchovávaní veľkého množstva dát. Súbory sa dajú: vytvoriť alebo vymazať, otvoriť alebo zatvoriť, čítať alebo zapisovať.

Podľa typu uchovávaných údajov (označované **príponou**), súbory rozdeľujeme na:

- **Textové súbory** - .txt, .csv, .html, .py
- **Obrazové súbory** - .bmp, .png, .jpg, .gif, .svg
- **Zvukové súbory** - .wav, .mp3, .midi
- **Video súbory** - .avi, .mp4, .mkv
- **Spustiteľné súbory** - .exe

V tejto kapitole budeme pre jednoduchosť pracovať s textovými súbormi.

#### 1. Prepisovanie

Príde ti zbytočne prepisovať dlhé články na vstup programu a vždy sa pomýliš. Načítaj články pre každú úlohu z predošlej kapitoly zo súboru. Uprav programy tak, aby si najprv vypýtali názov súboru. V úlohe „veľa opakovania“ ulož záznam o ceste robota do nového súboru.

## 2. Turistika

Na víkend sa črtajú ideálne podmienky na horskú turistiku. Nenecháš nič na náhodu a pripravíš si detailný plán s výškovým profilom trasy. Na každých desať metrov trasy si do súboru poznačíš nadmorskú výšku z mapy. Zisti celkové stúpanie a klesanie počas celého výletu spolu s najvyššou a najnižšou nadmorskou výškou. Vypíš aj celkovú dĺžku túry v kilometroch a trvanie prechodu horami v hodinách.

**Obsah súboru:**

```
348
351
379
384
395
401
396
```

**Vstup:**

Trasa je v súbore s názvom:

**Výstup:**

```
Trasa: 0.140 km - 0 h 21 min
Stúpanie: 53 m
Klesanie: 40 m
Najnižšie miesto trasy: 361 m
Najvyššie miesto trasy: 401 m
```

## 3. Vedomostný kvíz

Bifľovanie ti vôbec nepríde prínosné. Keby existoval spôsob, akým si opakovanie učiva spríjemniť. Včera si zo smútku nad vidinou takto premárneného času, pri jedení čokolády a čipsov, pozeral kvízovú reláciu. Prišlo ti to neuveriteľne poučné. Polož náhodnú otázku s možnosťami zo súboru kvízových otázok a bodovo ohodnoť správnu odpoveď. Všetky kvízové otázky s možnosťami sa však nezmestia do pamäti programu. Náhodnú otázku vyber priamo zo súboru.

**Obsah súboru:**

Otázka: V ktorom roku začala Francúzska revolúcia?  
 A: 1763  
 B: 1813  
 C: 1789  
 D: 1654  
 Odpoveď: C  
 Otázka: Al<sub>2</sub>O<sub>3</sub> je?  
 A: hydroxid vápenatý  
 B: oxid hlinitý  
 C: hydroxid sodný  
 Odpoveď: B

**Kvíz:**

Súbor s kvízovými otázkami:   
 Kvízové otázky pripravené. Ideme na to!  
 V ktorom roku sa začala Francúzska revolúcia?  
 A: 1763  
 B: 1813  
 C: 1789  
 D: 1654  
 Aká je správna odpoveď?:   
 Správne! Máš 1 bodov.  
 (alebo) Nabudúce si to lepšie premysli.

**4. Narodeniny**

Darčeky k narodeninám zvykneš kupovať na poslednú chvíľu. Potrebuješ mať prehľad aspoň na mesiac dopredu, kto bude mať narodeniny, aby si stihol vybrať niečo výnimočné. Zo súboru načítaj ľudí, ktorí majú sviatok v požadovaný mesiac v roku.

**Obsah súboru:**

Jožko Mrkvička, 15.3.2002  
 Katka Krátka, 2.7.1993  
 Martinko Klingáč, 12.11.1995  
 Iveta Novotná, 27.2.2001

**Vstup:**

Zobraz narodeniny pre mesiac v roku:



Výstup:

Narodeniny:

## 5. Cestovné poriadky

Z celoštátneho rýchlika prestupujú cestujúci v okresných mestách na miestne autobusy. Podľa času odchodu a trvania cesty zisti, ktorý autobus stihnú. Vypíš najbližší spoj s najmenším čakaním medzi vlakom a autobusom. Daj pozor! Prvý časový údaj v riadku s odchodom autobusu je trvanie cesty vlakom do stanice, odkiaľ odchádza ten autobus.

Obsah súboru:

vlak , 9:15 , 10:45 , 12:15 , 14:30 , 16:15 , 18:20  
bus , 1:00 , 11:00 , 13:00 , 15:00 , 17:00  
bus , 1:45 , 9:30 , 12:08 , 16:33

Vstup:

Čas:   
Trvanie cesty vlakom:

Výstup:

Najbližší spoje (vlak , autobus):

### 4.1.7 Funkcie

**Funkcia** je pomenovaná časť programu, ktorá vykonáva špecifickú činnosť. Hovorí sa im preto tiež **procedúry** alebo **podprogramy**. Predstavuje súvislú časť kódu. Blok kódu obsahuje sled na seba nadväzujúcich príkazov, ktorý tvorí jeden logický celok. Takto umožňuje zložitejší program rozdeliť na viacero samostatných častí.

#### 1. Vraký

V šírých hĺbinách Atlantiku sa stále ukrýva nepreberné bohatstvo vo vrakoch potopených lodí. V tejto minihre odkryješ tajomstvo skrývajúce sa pod hladinou. Cieľom je nájsť vrak parníka na náhodnej pozícii. Do programu napíš funkciu `vzdialenost(x, y)`, ktorá na základe zadaných súradníc vypočíta ako ďaleko si od vraku.

Vstup:

Sonar hlási potopený parník na dohľad!  
Tvoje súradnice?:

**Výstup:**

```
Od vraku si  námorných míľ.
...
Našiel si vrak. Dobrá práca!
```

## 2. Cézarová šifra

Na cestách po lodných pokladoch ťa odpočívajú piráti, ktorí ťa chcú predbehnúť a obohatiť sa. Na utajenie svojej polohy a správ s pevninou musíš informácie zašifrovať.

Funkcia `sifruj(sprava, kluc)` zašifruje text správy tak, že posunie každé písmeno abecedy podľa písmena `kluc`. Čiže správa “ABC” sa s kľúčom “B” zmení na “BCD”.

Funkcia `desifruj(sifra, kluc)` bude fungovať opačne. Pre lepšiu bezpečnosť podporuj aj dlhšie kľúče než len jedno písmeno. Každé písmeno bude potom vyjadrovať posun od začiatku abecedy písmena, s ktorým sa stretne. Správa “AVE CEZAR” s kľúčom “BCD” bude “BXH DGCBT”.

## 3. Pascalov trojuholník

Vytvor funkciu `pascalov_trojuholnik(n)`, ktorá vypíše súčtovú pyramídu s  $n$  riadkami. Pascalov trojuholník má po okrajoch samé jednotky. Ďalší riadok sa tvorí ako súčet dvoch susediacich čísel o riadok vyššie.

**Vstup:**

Počet riadkov:

**Výstup:**

```
    1
   1 1
  1 2 1
 1 3 3 1
1 4 6 4 1
```

## 4. Pekný byt

Investor musí poznať situáciu na trhu a potenciálnu konkurenciu predtým než si naplánuje stratégiu investovania. Rozbiehaš realitnú kanceláriu a skôr než stanovíš ceny pre byty v portfóliu, zisti v akom vzťahu je výmera bytu k jeho cene. Pre každú štatistiku napíš zodpovedajúcu funkciu. Údaje o bytoch načítaj zo súboru.

**Vstup:**Súbor s bytmi v lokalite: **Výstup:**

	: Cena (eur)	:	Výmera(m <sup>2</sup> )	:
Priemer	: <input type="text"/>	:	<input type="text"/>	:
Medián	: <input type="text"/>	:	<input type="text"/>	:
Modus	: <input type="text"/>	:	<input type="text"/>	:
Smerodajná odchýlka	: <input type="text"/>	:	<input type="text"/>	:

## 5. Rímske čísla

Od archeológov si dostal dlhý zoznam rímskych čísel. Nájdené boli v novoobjavených podzemných historických pamiatkach. Ťažko sa v nich dá vyznať a je na tebe, aby si ich premenil na „normálne“ arabské čísla. Poslali ti aj tabuľku pravidiel prevodu medzi rímskymi a arabskými ciframi. Napíš pre archeológov funkciu `rimске_na_arabske(rimske)`.

## 6. Základný tvar zlomku

Zlomky sú vhodné na presné výpočty s časťami z celku. Vytvor jednoduchú kalkulačku, ktorá umožňuje dva zlomky sčítať, odčítať, násobiť a deliť. Výsledok vždy zjednoduší na základný tvar.

**Vstup:**

Kalkulačka zlomkov

a =

b =

Vypočítaj (+, -, \*, /):

**Výstup:**

Výsledok:

## 7. Hra Poklad

Povráva sa, že na strašidelnom hrade v Karpatoch je bludisko so siedmimi tajomnými komnatami. Každá má meno a je v nej truhlica s pokladom. Mapa bludiska je náhodne poskladaná, uložená v pamäti počítača, ale nie je nakreslená na obrazovke. Hráč musí zistiť, ako sú komnaty navzájom pospájané. Na začiatku hry sa ocitne v náhodne vybranej komnate. Jeho úlohou je zhromaždiť všetky truhlice do spoločnej komnate. Môže však spraviť iba ohraničený počet krokov.

**Ukážka hry:**

```
Počítač rozumie týmto príkazom
S, V, J, Z   : Pohyb na sever, východ, juh, západ
ZDVIHNI     : Zdvihne truhlicu
POLOZ       : Položí truhlicu
KDE         : Informuje o polohe truhlic
SOS         : Vypíše pravidlá hry

Si v 4.komnate
Je žltá a žeravá
Sú v nej: ZLATKY
Čo chceš robiť?
? ZDVIHNI
Zdvihol si truhlicu, v ktorej sú zlatky.

Ešte stále si 4.komnate
Čo chceš robiť?
? Z
...
```

## 8. Kalkulačka

Moderné vedecké kalkulačky sú skoro zázrak. Buď tým, že sa mimo akademickej pôdy skoro vôbec nepoužívajú, alebo zložitou ich fungovaním. Dokážu rozlíšiť, či má prednosť násobenie alebo sčítanie, zatiaľ čo vezmú do úvahy zátvorky. Nemôže byť pre nich nič jednoduchšie ako prísť na to, čo je číslo a čo operátor. Vytvor program kalkulačky, ktorá sa bude správať ako vrecková vedecká kalkulačka, teda s infixovým zápisom.

**Ukážka**

**možností:**

```
> 5 * (1589 - 2 * 74) / 2 + (33 * 8)
> 3866.5
> ...
```

## 4.2 Vzorové riešenia

Riešenia úloh v učebniciach spravidla zachytávajú jedinú správnu odpoveď v podobe čísla, oznamovacej vety alebo nákresu bez vysvetlenia postupu. Možností korektného usporiadania príkazov v programe existuje v princípe neobmedzené množstvo, ktoré len vzrastá so znalosťami pokročilejších vyjadrovacích prostriedkov programovacieho

jazyka.

Podľa nášho názoru by ideálne riešenie úlohy podporujúce samoštúdium nemalo byť odhalené okamžite v celku, ale navádzať na postup cez čiastkové nápovede vo forme otázok. Riešiteľovi je za takých okolností jasnejšie, prečo si autor zvolili práve odkrytú programovú konštrukciu. Vyžaduje si to však náročnejšiu prípravu sady kontrolných otázok v učebnom materiáli. Alternatívne môže byť predostretý výpis programu zohľadňujúci požiadavky na vstup a výstup a oboznámenosť žiaka s témou v danom momente na základe predošlých úloh. V prípade ukázania už hotového riešenia slúži to iba na kontrolu. Pokladanie regulujúcich otázok spočíva na učiteľovi.

V **prílohe A** sa nachádzajú **exemplárne riešenia** takmer všetkých úloh zo zbierky okrem zameraných na opakovanie predošlého celku a na divergentné myslenie. U týchto cvičení je riešenie nanajvýš naznačené výpustkami (...), ktoré sú vyznačené v úlohe „6.1. Prepisovanie”. Spôsobom zachytenia riešení sa prikláňame v návrhu nášho učebného textu k použitiu na hodinách pod dohľadom učiteľa.

**Náznak riešenia:**

```
nazov_suboru = input("Názov súboru")
subor = open(nazov_suboru, "r")
for riadok in subor:
    riadok = riadok.strip()
    ...
subor.close()
```

Na vzorovom programe úlohy „3.3. Pyramída” ukážeme prispôbenie riešenia odporovaním chýb žiakov pri vypracovaní. Upozorníme na miesta, kde o zvolenom zápise rozhodujú predchádzajúce skúsenosti žiaka. Hneď prvý riadok, ktorý načítava vstup sa dá napísať tromi variantmi rastúcej zložitosti na porozumenie.

**Varianta č.1:**

```
print("Výška pyramídy: ")
vyska = input()
vyska = int(vyska)
```

**Varianta č.2:**

```
vyska = input("Výška pyramídy: ")
vyska = int(vyska)
```

**Varianta č.3:**

```
vyska = int(input("Výška pyramídy: "))
```

Variant č.1 prísne odlišuje medzi výstupom cez príkaz `print`, načítaním vstupu z klávesnice cez `input` a konverziou písmen na typ číslo `int`. Vo variante č.2 už operujeme s viacúčelovosťou príkazu `input`, ktorá sa v skorších fázach pletie s priradením konštantného reťazca do premennej. Najsofistikovanejší spôsob sa spolieha na zápis zloženej funkcie a predstavuje idióm jazyka. Žiadna syntax vzorového programu nie je univerzálna, ale vyvíja sa postupom kapitolami. V témach premenné a podmienky sa najčastejšie ukazuje variant č.2. Od témy cykly sa variant č.3 považuje za dostatočne známy.

**Násobenie reťazcov:**

```
for riadok in range(vyska):
    ...
    print(" " * medzery + "*" * hviezdy)
```

**Vnorené cykly:**

```
for riadok in range(vyska):
    ...
    for i in range(medzery):
        print(" ", end="")
    print()
    for i in range(hviezdy):
        print("*", end="")
    print()
```

Náročnosť úlohy a tým umiestnenie v systéme je ovplyvnené aj typmi príkazov vo vzorovom riešení. Tretia úloha v kapitole cykly má za cieľ precvičiť `for` cyklus s pevne určeným počtom opakovaní, preto sa medzery a hviezdičky vypisujú násobením reťazcov. Napriek tomu, že reťazce sa dosiaľ neprebrali do hĺbky, je jednoduchšie na pochopenie idea, že: „6 krát hviezdička vypíše 6 hviezdičiek” než mechanizmus za vnorenými `for` cyklami. Na tomto príklade sa ukazuje závislosť zaradenia úlohy od použitého jazyka, pretože nie všetky umožňujú násobenie reťazcov.

## 4.3 Elektronická učebnica

Dostupnosť aktualizovaného učebného textu odkiaľkoľvek umožňuje umiestnenie úloh s riešeniami do online prostredia<sup>1</sup>. Navigácia medzi zadaniami sa začína na klikateľnom

<sup>1</sup>Elektronická zbierka úloh: <https://etakerim.github.io/zbierka-uloh.html>

obsahu, ktorý vymenúva všetky ich názvy s poradovým značením.

Zobrazenie úloh na samostatných stránkach (Obr. 4.1) namiesto celej kapitoly na spoločnej stránke vyplýva zo snahy o zníženie nadbytočnej vonkajšej (*extraneous*) kognitívnej záťaže. Zo skúseností počas vyučovania nevedeli žiaci spoľahlivo nájsť požadovanú úlohu, keď sa nachádzali bezprostredne za sebou. Na orientáciu malo negatívny dopad aj číslovanie kapitol rímskymi číslami. Posúvanie medzi predošlou a nasledujúcou úlohou, rovnako aj vzorové riešenie sú prístupné hypertextovým odkazom umiestneným na spodku stránky.

#### 5. Pokazený rozpis

Továreň na železnú rudu dostala nový časový rozpis vylepšeného technologického procesu. Spracovanie zvyčajne trvá dlhšie ako hodinu. Nehodí sa im teda mať časy napísané iba v minútach. Rozpis programom minúty na dni, hodiny, minúty pre jednoduchšie čítanie rozpisu. Vynechaj nepotrebné časové údaje.

VSTUP:

Trvanie (min.): \_\_\_\_

VÝSTUP:

= \_\_\_\_ d. \_\_\_\_ hod. \_\_\_\_ min

- Na prehľad úloh
- ← Predchádzajúca úloha
- → Nasledujúca úloha

Vzorové riešenie

#### 5. Pokazený rozpis

```
min = input("Trvanie (min.): ")
min = int(min)

hod = min // 60
dni = hod // 24
hod -= dni * 24
min -= (hod * 60) + (dni * 24 * 60)

print("=", end=" ")
if dni > 0:
    print(f"(dni) d.", end=" ")
if hod > 0:
    print(f"(hod) hod.", end=" ")
print(f"(min) min.")
```

Späť na úlohu

Obr. 4.1: Zadanie a riešenie úlohy 2.5. na webe

Elektronická zbierka je voľná na stiahnutie cez portál *Github* pod licenciou *Creative Commons* s uvedením pôvodného zdroja a zachovaním licencie (*CC BY-SA 4.0*)<sup>2</sup>. V repozitári<sup>3</sup> cez systém na správu verzií *Git* je učebný text otvorený voči návrhom na kolaboratívne pridávanie nových úloh. Postačuje ak redaktor schváli zmenu prichádzajúcu cez *Pull request*. Webová stránka zbierky úloh sa potom automaticky zverejní prostredníctvom *Github pages*<sup>4</sup>.

## 4.4 Diskusia

Rozbor učebného textu tvoreného problémovými úlohami zo základov programovania demonštrujeme vo dvoch líniách. Vyjadrujeme jednak číselné charakteristiky textov zadanií a takisto zovšeobecňujeme ich ustálenú obsahovú podobu. Rámec na budúce obohacovanie zbierky o nové zadania sa opiera nemej o výrazové prvky programovacieho

<sup>2</sup>CC BY-SA 4.0 licencia: <https://creativecommons.org/licenses/by-sa/4.0/deed.cs>

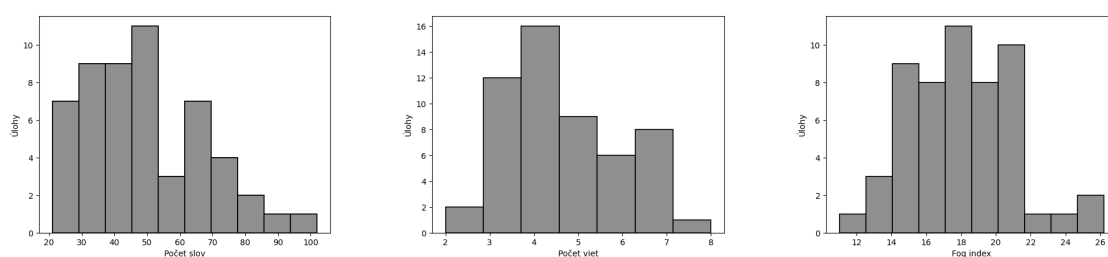
<sup>3</sup>Git repozitár zbierky úloh: <https://github.com/etakerim/etakerim.github.io>

<sup>4</sup>Projekt Github pages: <https://pages.github.com/>

jazyka, ktoré má úloha precvičovať, a následnú kategorizáciu úlohy v tematickom celku ku vzťahu k existujúcim úlohám.

#### 4.4.1 Číselný opis textov zadání

Združenia viet popisujúce problémovú situáciu a pokyny k činnosti patriace pod spoločnú úlohu boli vyhodnotené algoritmom uvedeným v prílohe C pre každú úlohu zvlášť. Ako číselné vlastnosti sme zvolili počet slov, počet viet a skóre čitateľnosti cez fog index.



Obr. 4.2: Kvantifikatívne vlastnosti textu úloh

V prílohe B sú získané hodnoty v tabuľkách pre všetky znenia cvičení zoskupené po kapitolách. Histogramy (4.2) poukazujú na to, že priemerný počet slov naprieč zadaniami je  $49 \pm 18$  a teda dominujú kratšie texty v rozmedzí 30 až 80 slov. Priemerný počet viet je  $4 \pm 1$  a žiaden text ich nemá viac ako 8. Fog index sa sústreďuje okolo skóre  $18 \pm 3$ , čo značí že texty sú primerane zrozumiteľné pre žiakov stredných škôl. Vo všeobecnosti texty so skóre pod 20 sa pokladajú za veľmi ľahko čitateľné. Do vzorca fog indexu vstupuje počet slov na vetu, ktorých priemer je  $10 \pm 2$ , a percento ťažkých slov zo všetkých, tých je priemerne  $33.9\% \pm 8\%$ .

Merania fog indexu sme brali výhradne ako odporúčenie na úpravu textu, vzhľadom na prispôsobenie anglickým textom kritériom pre náročné slová a optimálnu dĺžku textu 100 slov. Napriek tomu sme niektoré cvičenia so značne vyšším fog indexom uspôsobili skrátením komplikovaním viet a zamenením náročnejších slov za bežné. Hodnovernejšie posúdenie by poskytlo sémantické kritérium špeciálne určené pre úlohy, také nám lež nie je známe.



### 4.4.2 Predloha štruktúry úlohy

Aby podoba znení pôsobila rovnorodým dojmom pridržujeme sa stáleho rozvrhnutia vnútorného usporiadania textu cvičenia. Zadanie sa začína opisom scenérie prostredia, kde sa dej odohráva. Nasledujú fakty, vzťahy a požiadavky na vstupy uvádzajúce algoritmizovateľnú problémovú situáciu. Úryvok ukončuje najčastejšie explicitné vyjadrenie pokynu v rozkazovacom spôsobe. Pokyn na riešenie nemusí byť priamo vyslovený. Na objasnenie očakávaní techník, ktoré má žiak zvoliť pri riešení, slúži ukážka behu programu.

Prítomnosť opísaných štruktúrnych prvkov textu dokladáme vybranými úlohami zo zbierky. Tri hlavné časti nazývame skráteno: **scenéria**, **problém** a **pokyn**.

### 1.5. Hlboká roklina

(**Scenéria**): „Stojíš nad hlbokým údolím za zábradlím a uvažuješ ako odmerať jeho hĺbku. Vtom si spomenieš na svoje vedomosti z fyziky. Zoberieš si do ruky povalúci sa kameň a pustíš ho priamo do rokliny.”

(**Problém**): Zároveň stlačíš stopky, ktorými zmeriaš čas do dopadu v sekundách. Kameň padá nadol voľným pádom. Stopky zastavíš pri započutí rachotu z nárazu. Pri výpočte zanedbáme rýchlosť zvuku, ktorou sa rachot rozšíri až k nám.”

(**Pokyn**): Implicitný

### 2.1. Heslo

(**Scenéria**): „Tvoj dom na strome už vykradlo pár nezvaných návštevníkov. Vymyslel si preto spôsob ako povoliť návštevu len overeným osobám.”

(**Problém**): „Tie musia poznať tajné heslo.”

(**Pokyn**): „Napíš program, ktorý slovne privíta členov a odoženie zlodějov.”

### 2.4. Morský vánok

(**Scenéria**): „Kapitán plachetnice na otvorenom oceáne musí mať vždy prehľad odkiaľ fúka vietor, aby odkormidloval do vytúženého cieľa. Príliš silné závaný vetra môžu byť nebezpečné pre posádku. Polámať lodné sťažne, potrať plachty, či zaplaviť palubu.”

(**Problém**): „Cez rádio dostáva plavidlo každý deň správy o predpovedi sily vetra v Beafortovej stupnici. Sila vetra je ňou vyjadrená do dvanástich stupňov od bezvetria až po orkán.”

(**Pokyn**): „Napíš program, ktorý kapitánovi vysvetlí stupeň vetra. Podľa stupnice určíme jeho pomenovanie, rýchlosti v námorných uzloch a očakávanej výšky vln.”

### 5.5. Výskyt písmen

**(Scenária):** „Dlho do noci čítaš časopisy o umelej inteligencii a fascinuje ťa jej schopnosť rozprávať sa s človekom.”

**(Problém):** „Na vytvorenie viet na danú tému potrebuje mať prehľad o percentuálnom výskyte hlások v texte.”

**(Pokyn):** „Spočítaj a vypíš zoznam početnosti písmen v reťazci.”

### 7.1. Vraký

**(Scenária):** „V šírých hĺbinách Atlantiku sa stále ukrýva nepreberné bohatstvo vo vrakoch potopených lodí. V tejto minihre odkryješ tajomstvo skrývajúce sa pod hladinou.”

**(Problém):** „Cieľom je nájsť vrak parníka na náhodnej pozícii.”

**(Pokyn):** „Do programu napíš funkciu vzdialenosť( $x$ ,  $y$ ), ktorá na základe zadaných súradníc vypočíta ako ďaleko si od vraku.”

Písomný prejav má tvar 2. osoby jednotného čísla, čím si kladieme predpoklad, že zlepšíme zrozumiteľnosť, keď sa priblížime reči rovesníkov. Uľahčenie v chápaní komunikačného zámeru dosiahnutý pravidelnosťou, nám umožníme pridať na inferečnej zložitosti. Namiesto žiakovho sústredenia sa na povrchovú orientáciu v texte ponecháme priestor na vyvodzovanie súvislostí, čím podporujeme čítanie s porozumením.

Grafická úprava úlohy je taktiež zjednotená a sa skladá z nadpisu, textu zadania, ukážka vzorového súboru ak je prítomná, vstupov a výstupov programu. Nadpis úlohy obsahuje číselné označenie kapitoly, poradia úlohy vnútri kapitoly a jej názov v podobe slovného spojenia. Vstupy a výstupy programu sú orámované v oblasti so šedým pozadím, písmo je neproporcionálne, a premenlivé údaje závislé od vstupov sú zastúpené prázdnyimi obdĺžnikmi. Na ilustráciu správania po spustení programu sú do obdĺžnikov niekedy vpísané konkrétne hodnoty.

#### 4.4.3 Didaktické funkcie úloh

Graduálne stupňovanie náročnosti úloh do troch úrovní, rovnomerné zastúpenie napĺňania rozličných didaktických funkcií a zapojenie viacerých kognitívnych úrovní overíme klasifikovaním úloh. Príloha B zachytáva zaradenie samotných zadaní do systému úloh. Tému, podtému a element sú prevažne predurčené názvom kapitoly, preto tie nešpecifikujeme.

**Určenie didaktických funkcií** úlohy je do značnej miery subjektívne, preto na odôvodnenie našich rozhodnutí ponúkame osobitnú interpretáciu kategórií. Logickým štruktúrovaným učiva tematickej oblasti postupne žiaci prechádzajú istými fázami v oboznamovaní sa s učebnou látkou, ktoré podporujú najlepšie úlohy s určitou didaktickou funkciou.

**Prípravné úlohy** sa vyskytujú pred uvedením samotného elementu učiva a majú navádzať k potrebe naučenia sa novej metódy, ktorá je v zapätí akoby objavená. Javy z programovania, kde k tomu dochádza, sú duplikácia rovnakého kódu tesne pred predstavením cyklov alebo výskyt veľa premenných základného typu pred oboznámením sa s jednorozmerným poľom. V našej zbierke sa také úlohy priamo nenachádzajú, ale ich znaky pripisujeme niektorým úvodným úlohám kapitol.

Po prípravných úlohách nastupujú na rad **propedeutické úlohy**, s ktorými sa zvyknú prelínať. Propedeutické úlohy však poskytnú triviálny prípad použitia novej syntaxe, a nepoukazujú už na ťažkopádnosť doposiaľ naučených možností zápisu problému. Zložitosťou majú byť tieto skupiny úloh menej náročné.

Sekvencia etáp vyučovacieho procesu predraďuje motiváciu pred expozíciou. V našom ponímaní majú úlohy s **motivačnou funkciou** vzhľadom k poznávaciemu procesu obrazne donútiť žiaka, aby preskúmal externé zdroje informácií. Fakty, ktoré nenájde v učebnici programovania, má byť potom motivovaný hľadať v učebniciach iných predmetov, v príručkách, knihách alebo na internete. Týkať sa to môže napr. fungovania hodu kockou, nájdenia matematických vzťahov, či dopĺňania banky otázok pre kvíz.

Po uvedení cez propedeutické úlohy sa prepracúvame k úlohám na **osvojenie pojmu, vzťahov a postupu**. Od prvotného vystaveniu sa koncept prvku programovacieho jazyka sa ukazuje v tej istej základnej podobe v rozličných situáciách. Spomenieme cyklus, ktorý vypíše najprv desať totožných riadkov, potom hviezdičky v rovnakom riadku alebo hviezdičky po diagonále. Ukončovacia podmienka for cyklu zostáva zakaždým rovnaká, pribúdajú výlučne príkazy bez akéhokoľvek ovplyvňovania toku programu v tele cyklu. Úlohy na osvojovanie sú menej náročné.

Úlohy naplňajúce funkciu **upevňovania učiva** sa málo líšia od **úloh na opakovanie a systematizáciu**. Zámena účelu úlohy z týchto dvoch skupín závisí kedy ju žiak bude riešiť vzhľadom na doterajšie nadobudnuté znalosti a skúsenosti. Pokiaľ sa pridržiava nami vymedzeného poradia úloh, tak funkcia upevňovania prislúcha sko-

rej uvedeným úlohám, kde ešte dochádza k postupnému nabaľovaniu vyjadrovacích možností príkazu, podmienka *if* sa rozširuje o vetvy *elif* a *else*. Systematizáciou sa nepridávajú vlastnosti jazyka, ale stúpa náročnosť problémových situácií, na ktoré sú aplikované. Tvorí predpoklady pre stredné aj vysoko náročné úlohy.

**Aktualizačné úlohy** zapájajú skorej osvojené učivo do inovatívnych situácií, kde nastáva po prvýkrát súhra medzi staršími a novými elementami programovacieho jazyka. Najskoršie umiestnená úloha modelovo napríklad na vetvenie v cykle alebo na iteráciu cez znaky reťazca namiesto cez inkrementujúce sa počítadlo sú úlohy s aktualizacnou funkciou.

**Aplikačná funkcia mimo alebo vo vnútri** informatiky súvisí s medzipredmetovými vzťahmi, ktoré rozoberáme v ďalšej časti. Podstata odhalenia aplikácie v inom vyučovacom predmete vychádza z pojmov v texte zadania cvičenia.

Zoradenie systému úloh v rámci témy zbierky sa riadi ich najmarkantnejšou didaktickou funkciou. Od jednoduchším k zložitejším zadaniam začína prípravnou funkciou a pokračuje propedeutikou, osvojovaním, upevňovaním, opakovaním a systemizáciou. Priebežne celou témou sa ťahne aktualizácia predošlého učiva a motivácia. Kde nastane zhoda v hlavnej funkcii, tam sa skôr umiestni úloha s nižšou úrovňou vyžadovaných poznávacích procesov. Úlohy na konvergentné myslenie predchádzajú tie s divergentným riešením.

#### 4.4.4 Medzipredmetové vzťahy

Podľa didaktickej zásady spojenia teórie s praxou a na príspevok ku kompetenciám celoživotného učenia sa pre život dbáme na interdisciplinaritu nastolených problémov zasadených do reálneho sveta. Medzipredmetové aplikácie čerpajú z matematiky (16 úloh) a finančnej gramotnosti (2 úlohy), fyziky (4 úlohy), slovenského jazyka a literatúry (3 úlohy), geografie (2 úlohy), dejepisu (2 úlohy) a chémie (1 úloha).

Matematika má z hľadiska využitia jej aparátu v programátorských úlohách nespornú výhodu pre blízkosť k informatike ako takej. Existujú zároveň viaceré možnosti znovupoužitia príkladov z matematických učebníc. K medzipredmetovým vzťahom s **matematikou** prispievajú témy: objem a obsah telies (1.6, 1.8, 1.9), meranie času (2.5), kalkulačky (2.6, 7.8), kvadratické rovnice (2.8), trojuholníky (2.9), malá násobilka (3.8, 4.3), zlomky a percentá (5.5, 5.6, 7.6), finančná gramotnosť (3.9, 5.7), súradni-

cová sústava (7.1), binomická veta (7.3), štatistika (7.4). Do **fyziky** môžeme zaradiť úlohy ohľadom: teploty (1.4), kinematiky (1.5, 1.7), dynamiky (1.9). **Slovenského jazyka** sa okrajovo týkajú úlohy na: rým (1.2), hláskoslovie (5.2), tvaroslovie (5.8). Pod témy súvisiace s **geografiou** spadajú meteorológia (2.4) a kartografia (6.2), z **chémie** sme zahrnuli roztoky (1.10) a z **dejepisu** sa zdieňujeme o Májskych pyramidach (3.3) a rímskych číslach (7.6).

V zbierke je tiež 9 úloh s rýdzou **informatickým** zameraním: heslá (2.1), hry a simulácie (4.1, 4.2, 7.7), šifrovanie (7.2), kompresia (5.9), databázy (6.3, 6.4, 6.5).

Výber vhodných úloh spolieha na učiteľa, aby pružne reagoval na časové súvislosti v učebných plánoch nadväzujúcich predmetov. Predbiehať predmetu, z ktorého úloha čerpá, je viac nevyžiadané než oneskorenie, pretože to so sebou nesie znásobené časové nároky na objasnenie a hrozbu nadobudnutia povrchného obrazu o prebratej oblasti z iného vyučovacieho predmetu.

## 5 Záver

V záverečnej práci sme poukázali na problém s nedostatkom kvalitných učebníc v stredoškolskej informatike a problémových cvičení, ktoré by podnecovali objavnú tvorivú činnosť žiakov a umožňovali domácu prípravu. Ako reakciu na tieto výzvy sme vytvorili súbor 54 slovných úloh na programovanie so vzorovými riešeniami v jazyku Python.

Učebnou náplňou sa v zbierke dohromady presahujú požiadavky na základné učivo vymedzené obsahovým a výkonovým štandardom štátneho vzdelávacieho programu. Rozsahu vzdelávacích štandardov pre algoritmizáciu a programovanie sa venujú kapitoly z našej zbierky, hlavne: premenné, podmienky, cykly. Pre školy s rozšíreným ŠkVP informatiky oproti ŠVP a pre maturantov z informatiky sú určené kapitoly: náhodné čísla, reťazce a zoznamy, súbory, funkcie.

V druhej časti čerpajúcej z cieľových požiadaviek na maturitu z informatiky si väčšmi uvedomujeme nepostačujúci počet zadaní s propedeutickou funkciou a tých na osvojovanie novozavedenej syntaxe jazyka. V časti funkcie sme identifikovali veľa divergentných úloh na úkor jednoduchých úloh na systemizáciu, ktoré by boli prístupné aj slabším žiakom. Na zaplätanie spomenutých nedostatkov v rozmanitosti do budúcnosti, sme predstavili metódu na zaraďovanie ďalších úloh. Metóda spočíva v zhodnotení náročnosti úlohy podľa jej dominantných didaktických funkcií, požadovanej poznávacej úrovne a prvkov jazyka vyskytujúcich sa vo vzorovom riešení programu.

Na zjednotenie predstáv potenciálnych spoluautorov sme vypracovali interpretačný popis pre didaktické funkcie s modelovými prípadmi ako podobné skupiny navzájom odlišiť. Podľa stúpajúcej náročnosti sme dospeli k nasledovnému poradiu funkcií: prípravná, propedeutická, osvojovacia, upevňovacia, opakovanie a systematizácia. Motivačné úlohy a úlohy na aktualizáciu predošlého učiva majú byť zaraďované priebežne. Navyše sme dosiahli zahrnutie medzipredmetových vzťahov v 56% úloh z celkového počtu. Aplikáciou sú zasadené do kontextu vyučovacích predmetov: matematika, fyzika, slovenský jazyk a literatúra, geografia, chémia, dejepis.

Od úloh požadujeme šablónovitú príbehovú zmenu zadania, ktoré sa člení

do troch častí: scenéria, problém a pokyn. Podľa nami vytvorených úloh vyplýva, že úloha sú krátke texty priemerne majú  $49 \pm 18$  slov,  $4 \pm 1$  viet, s  $10 \pm 2$  slovami na vetu a  $33.9\% \pm 8\%$  slovami s dvoma a viac slabikami. Skóre čitateľnosti Gunning fog index je  $18 \pm 3$  a ideálne sa má udržiavať pod hodnotou 20. Zbierka je náročnosťou textu prístupná žiakom stredných škôl s menšími výhradami.

Grafickej úprave zbierky tiež predpisujeme jednotnú podobu s číslovaným názvom úlohy, textom znenia problémovej situácie, ukážkou súboru, vstupu a výstupu programu. V online podobe učebného textu sa pod špecifikáciu úlohy umiestňujú ešte navigačné hypertextové prepojenia. Učebný text je prístupný na webovej stránke a cez portál Github podporuje kolaboratívnu modifikáciu. Rozvrhnutie materiálu v online prostredí sa podriaďuje osvedčeným metódam na znižovanie vonkajšej kognitívnej záťaže ako je zjednodušenie štruktúry obsahu a jeho segmentácia kapitoly.

Súčasná práca poskytuje možnosti na viaceré rozšírenia. Vzorové riešenia úloh by sa po vzore programového vyučovania mohli transformovať na postupnosť napovedajúcich otázok a odpovedí v podobe kusov programového kódu. Ďalšie problémy sa naskytajú v súvislosti s detailnejším overením integrácie novovytvorených úloh do zbierky podľa vypracovanej metódy. Porovnanie úspešnosti žiakov pri práci s papierovou verzus elektronickou učebnicou využívajúcou hypertext nám zas umožní lepšie porovnať prínosy týchto prezentačných médií.

# Bibliografia

- BLAHO, Andrej; SALANCI, Ľubomír, 2019a. *abcPython - Pracovné listy pre Python*. Dostupné tiež z: <https://abcpython.input.sk/>.
- BLAHO, Andrej; SALANCI, Ľubomír, 2019b. *Metodiky k pracovným listom pre Python*. Dostupné tiež z: <https://abcpython-metodika.input.sk/>.
- BUŠEK, Ivan; ŠEDIVÝ, Jaroslav; MANNOVÁ, Božena; RIEČAN, Beloslav, 1987. *Zbierka úloh pre 3. ročník gymnázia*. 1. vyd. Bratislava: Slovenské pedagogické nakladateľstvo. ISBN 67-169-87.
- DRAHOŠOVÁ, Renáta, 2014. *Hodnotenie zrozumiteľnosti textu učebnice*. Brno. Diplomová práca. Pedagogická fakulta, Masarykova univerzita.
- GAVORA, Peter, 1992. *Žiak a text*. Bratislava: Slovenské pedagogické nakladateľstvo. ISBN 80-08-00333-2.
- HEDVIGOVÁ, Mária, 2007. *Algoritmy a programovanie v Pascale: nielen pre maturantov z predmetu informatika*. Dostupné tiež z: <http://web.archive.org/web/20200218113709/http://www.programovanie.kromsat.sk/prog-b/index.htm>.
- KROTKÝ, Jan, 2015. *Nové formy tvorby multimediálnych učebníc*. Plzeň. Dizertačná práca. Západočeská univerzita v Plzni.
- KSP, 2022. *Kuchárka KSP - Korešpondenčný seminár z programovania*. Dostupné tiež z: <https://www.ksp.sk/kucharka/>.
- KUČERA, Peter, 2016. *Programujeme v Pythone: učebnica informatiky pre stredné školy*. 1. vyd. Bratislava. ISBN 978-80-972320-4-7.
- MÉSZÁROSOVÁ, Eva, 2017. *Python a korytnačia grafika*. Bratislava: Knižničné a edičné centrum FMFI UK.
- MINĎÁKOVÁ, Ingrid, 2008. *Tvorba systémov úloh a ich implementácia do zbierky úloh*. Košice. Autoreferát dizertačnej práce. Univerzita P.J.Šafárika.



- MINEDU, 2019. *Cieľové požiadavky na vedomosti a zručnosti maturantov z informatiky*. Bratislava: Štátny pedagogický ústav. Dostupné tiež z: [www.statpedu.sk](http://www.statpedu.sk).
- MINEDU, 2023. *Štátny vzdelávací program pre gymnáziá v Slovenskej republike*. Bratislava: Štátny pedagogický ústav. Dostupné tiež z: [www.statpedu.sk](http://www.statpedu.sk).
- MISHRA, Sanjaya; SHARMA, Ramesh C. (ed.), 2005. *Interactive multimedia in education and training*. Hershey [Pa.]: Idea Group Pub. ISBN 978-1-59140-393-7.
- MLADÝ, Karol, 1988. *Tvorba a výroba učebníc*. 1. vyd. Bratislava: Slovenské pedagogické nakladateľstvo.
- NRSR, 2023. Zákon č. 245/2008 Z. z. o výchove a vzdelávaní (školský zákon) a o zmene a doplnení niektorých zákonov. *Zbierka zákonov*.
- PAVLOVKIN, Michal; MACKOVÁ, Zdenka, 1989. *Žiak a učebnica: Psychologické východiska tvorby učebníc pre mladších žiakov*. 1. vyd. Bratislava: Slovenské pedagogické nakladateľstvo. ISBN 80-08-00109-7.
- ŠEDIVÝ, Jaroslav; BOCKO, Vladimír; BOČEK, Leo; MANNOVÁ, Božena; MÜLLEROVÁ, Jana; POLÁK, Josef; RIEČAN, Beloslav, 1986. *Matematika pre 3. ročník gymnázia*. 1. vyd. Bratislava: Slovenské pedagogické nakladateľstvo.
- ŠVEDA, Dušan, 1992. *Tvorba systémov úloh v matematike*. Prešov: MC.
- TATCHELLOVÁ, Judy; BENNET, Bill, 1990. *Skúsiš to s mikropočítačom?: Poznávame a programujeme*. Bratislava: Mladé letá. ISBN 80-06-00107-3.
- UHERČÍK, Milan, 2012. *Význam priestorovej konceptualizácie pre znižovanie kognitívnej záťaže v grafickom užívateľskom prostredí*. Bratislava. Diplomová práca. FMFI Univerzita Komenského.
- WATTSOVÁ, Lisa; WALTERS, Gaby, 1991. *Skúsiš programovať?: Basic a strojový kód*. Bratislava: Mladé letá. ISBN 80-06-00178-2.
- ZELINA, Miron, 1990. *Tvorivosť v matematike (metodický materiál)*. Bratislava.
- ZUJEV, Dmitrij Dmitrijevič, 1986. *Ako tvoriť učebnice*. 1. vyd. Bratislava: Slovenské pedagogické nakladateľstvo.

# Príloha A: Kompletné riešenia úloh

## A.0.1 Premenné

### 1. Pozdrav

```
meno = input("Ako sa voláš?: ")
print("Ahoj ", meno)
print("Dovidenia ", meno)
```

### 2. Básnik

```
slovo = input("Napiš slovo, ktoré sa rýmuje so slovom strach: ")
print("Tu je báseň:")
print("Z počítačov mával som vždy strach")
print("teraz som však šťastný ako", slovo, ".")
```

### 3. Pozvánka

```
meno = input("Meno kamaráta: ")
cas = input("Čas oslavy: ")
vec = input("Prinesie okrem darčeku: ")

print(f"Ahoj {meno},")
print(f"pozývam ťa na moju narodeninovú párty.")
print(f"Bude sa konať 12.4. o {cas}.")
print("Nezabudni priniesť {vec} a pekný darček.")
print("Teším sa na teba! :)")
```

### 4. Teplota vo Farenheitoch

```
f = input("Vonku je °F: ")
f = float(f)
c = (5 / 9) * (f - 32)
print(f"Doma by bolo na teplomeri {c:.2f}°C.")
```

## 5. Hlboká rokľina

```
g = 9.81
t = input("Čas do dopadu kameňa: ")
t = int(t)
h = (g * (t ** 2)) / 2
print("Hĺbka rokľiny je", h, "metrov")
```

## 6. Vedro s vodou

```
pi = 3.14159
v = input("Výška vedra (cm): ")
d = input("Priemer dna (cm): ")
v = int(v)
d = int(d)
V = pi * ((d / 2) ** 2)
V = V / 1000
print("Do vedra sa zmestí", V, "litrov vody.")
```

## 7. Cesta autom

```
km = input("Dĺžka cesty (km): ")
odchod = input("Odchod z domu (hodina): ")
prichod = input("Príchod do hotela (hodina): ")

km = float(km)
odchod = int(odchod)
prichod = int(prichod)
hod = prichod - odchod

print(f"Auto pôjde priemernou rýchlosťou {km / hod:.2f} km/h.")
```

## 8. Kúpalisko

```
dlzka = input("Dĺžka bazéna (m): ")
sirka = input("Šírka bazéna (m): ")
hlbka = input("Hĺbka bazéna (m): ")
okraj = input("Hĺbka hladiny od okraja (cm): ")
cena = input("Cena za m^3 vody v eurách: ")
```

```
dlzka = float(dlzka)
sirka = float(sirka)
hlbka = float(hlbka)
okraj = int(okraj)
cena = float(cena)
V = dlzka * sirka * (hlbka - (okraj / 100))
V *= 1000
cena = cena * V

print(f"Na bazén sa minie {V} litrov vody")
print(f"Voda bude to stáť {cena} eur.")
```

## 9. Maľovanie

```
# Získaj z klávesnice rozmery miestnosti
print("Rozmery miestnosti")
dlzka = input("Dĺžka (cm): ")
sirka = input("Širka (cm): ")
vyska = input("Výška (cm): ")

# Premeň z písmen na čísla
dlzka = int(dlzka)
sirka = int(sirka)
vyska = int(vyska)

# Získaj z klávesnice rozmery okna a výdatnosť farby
print("Rozmery okna")
sirkaOkna = input("Širka (cm): ")
vyskaOkna = input("Výška (cm): ")
vydatnost = input("Výdatnosť farby (m^2/kg): ")

# Premeň z písmen na čísla
sirkaOkna = int(sirkaOkna)
vyskaOkna = int(vyskaOkna)
vydatnost = float(vydatnost)

# Spočítaj plochy stien, stropu a odpočítaj plochu okna
PlochaMiestnost = \
```

```
(dlzka * sirka) + 2 * (vyska * sirka) + 2 * (vyska * dlzka)
PlochaOkno = sirkaOkna * vyskaOkna
S = (PlochaMiestnost - PlochaOkno) / 10000
farbaKg = S / vydatnost

print(f"Maľovať budeš plochu {S:.2f} m2.")
print(f"Kúp {farbaKg:.2f} kg farby.")
```

## 10. Chemikálie

```
m1 = int(input("Hmotnosť roztoku č.1 (m1)?"))
w1 = int(input("Hmotnostný zlomok roztoku č.1 (w1)?"))
m2 = int(input("Hmotnosť roztoku č.2 (m2)?"))
w2 = int(input("Hmotnostný zlomok roztoku č.2 (w2)?"))

m3 = m1 + m2
w3 = (m1 * w1 + m2 * w2) / m3

print(f"Výsledný roztok má hmotnosť", m3, "g")
print(f"Hmotnostný zlomok rozpustenej látky je", w3 * 100, "%")
```

## 11. Brzdzenie

```
import math
print("Vlaková súprava")
v = int(input("- Rýchlosť (km/h): "))
lokomotiva = float(input("- Hmotnosť lokomotívy (t): "))
vagon = float(input("- Hmotnosť vagóna (t): "))
pocet_vagonov = int(input("- Počet vagónov: "))
F_b = int(input("- Brzdná sila (N/t): "))

# Premeň jednotky na základné SI
v /= 3.6
lokomotiva *= 1000
vagon *= 1000
F_b /= 1000

# Hmotnosť súpravy je hmotnosť lokomotívy a všetkých vagónov
m = lokomotiva + (pocet_vagonov * vagon)
```

---

```

# Vypočítaj celkovú kinetickú energiu, tá je rovnaká ako práca
# ktorú musia brzdy vykonať na zabrzdzenie.
W = 0.5 * m * (v ** 2)
# Celková sila pôsobiaca proti pohybu vlaku
F = F_b * m
# Z definície práce W = F * s, vypočítaj dráhu potrebnú na zastavenie
s = W / F
# Vypočítaj čas potrebný na zastavenie pre rovnomerný spomalený pohyb
a = F / m
t = math.sqrt(2 * s / a)

print(f"Vlaková súprava má hmotnosť {int(m / 1000)} ton.")
print(f"V rýchlosti {int(v * 3.6)} km/h zabrzdí na vzdialenosť {int(s)} metrov.")
print(f"Brzdzenie bude trvať {int(t)} sekúnd.")

```

## A.0.2 Podmienky

### 1. Heslo

```

print("Stoj! Povedz Heslo!")
pokus = input("? ")
if pokus == "tajne heslo":
    print("Vstúp, priateľ")
else:
    print("Zmizni kade ľahšie")

```

### 2. Najväčšie číslo

```

x = input("1.skóre: ")
y = input("2.skóre: ")
z = input("3.skóre: ")
x = int(x)
y = int(y)
z = int(z)
najviac = x
poradie = 1
if y > najviac:
    najviac = y

```

```
    poradie = 2
if z > najviac:
    najviac = z
    poradie = 3

print(f"Najväčšie skóre {najviac} bodov má {poradie} hráč.")
```

### 3. Vhodné oblečenie

```
pocasio = input("Ako je vonku?: ")
miesto = input("Kam ideš?: ")

if pocasio == "slnečno"
    povinne = "šiltovka"
if pocasio == "zamračené":
    povinne = "mikina"
if pocasio == "dážď":
    povinne = "vetrovka"

if miesto == "ihrisko":
    odporucanie = "tepláky"
if miesto == "škola"
    odporucanie = "košela"

print("Určite si nezabudni", povinne, "a tiež si vezmi", odporucanie, ".")
)
```

### 4. Morský vánok

```
stupen = input("Sila vetra na Beaufortovej stupnici: ")
stupen = int(stupen)

if stupen == 0:
    nazov = "bezvetrie"
    rychlost = 0
    vlny = 0
elif stupen == 1:
    nazov = "vánok"
    rychlost = 2
```

```

    vlny = 0.1
elif stupen == 2:
    nazov = "slabý vietor"
    rychlost = 5
    vlny = 0.2
# Doplň ostatné stupne podľa Beafortovej stupnice

print(f"Vietor sa nazýva {nazov}.")
print(f"Vietor má rýchlosť {rychlost} kt.")
print(f"Očakávaná výška vĺn je {vlny} m.")

```

## 5. Pokazený rozpis

```

min = input("Trvanie (min.): ")
min = int(min)
hod = min // 60
dni = hod // 24
hod -= dni * 24
min -= (hod * 60) + (dni * 24 * 60)

print("=", end=" ")
if dni > 0:
    print(f"{dni} d.", end=" ")
if hod > 0:
    print(f"{hod} hod.", end=" ")

print(f"{min} min.")

```

## 6. Hovoriaca kalkulačka

```

print("Som hovoriaca kalkulačka a rada počítam!")
a = int(input("Povedz mi prvé číslo: "))
b = int(input("Potrebujem ďalšie číslo: "))
cinnost = input("Chceš ich sčítať alebo odčítať: ")

if cinnost == "sčítať":
    print(f"Výsledok tvojho príkladu: {a} plus {b} je {a + b}")
elif cinnost == "odčítať":
    print(f"Výsledok tvojho príkladu: {a} mínus {b} je {a - b}")

```



```
else:
    print(f"Neviem čo znamená '{cinnost}'")
```

## 7. Chaos v lístkoch

```
print("Popíš mi svoju cestu s MHD")
zony = int(input("Koľko zón prejdeš?:"))
minuty = int(input("Koľko minút má trvať cesta?:"))

if zony == 2 and minuti <= 30:
    cena = 0.55
elif zony == 3 and minuti <= 60:
    cena = 0.80
elif zony == 4 and minuti <= 60:
    cena = 1.00
elif zony == 5 and minuti <= 90:
    cena = 1.25
elif zony == 6 and minuti <= 90:
    cena = 1.50
elif zony == 7 and minuti <= 120:
    cena = 1.65

print("Zlavnený lístok stojí {cena:.2f} eur.")
```

## 8. Kvadratická rovnica

```
import math
print("Koeficienty kvadratickej rovnice:")
a = float(input("a = "))
b = float(input("b = "))
c = float(input("c = "))

if a == 0:
    print("Ide o lineárnu rovnicu")
else:
    print(f"{a:g}x^2 + {b:g}x + {c:g} = 0")
    D = b ** 2 - 4 * a * c
    if D < 0:
        print("Kvadratická rovnica nemá riešenie v R")
```

---

```

elif D > 0:
    x1 = (-b - math.sqrt(D)) / (2 * a)
    x2 = (-b + math.sqrt(D)) / (2 * a)
    print(f"x1 = {x1}")
    print(f"x2 = {x2}")
elif D == 0:
    x = -b / (2 * a)
    print(f"x = {x}")
    Vx = -b / (2 * a)
    Vy = c - ((b ** 2) / (4 * a))
    print(f"V[{Vx}]; {Vy}")

```

## 9. Trojuholníky

```

import math

print("Zadajte strany ľubovoľného trojuholníka:")
a = input("a = ")
b = input("b = ")
c = input("c = ")

a = float(a)
b = float(b)
c = float(c)

if a + b <= c:
    print("Pre trojuholník neplatí trojuholníková nerovnosť")
    print("a + b <= c")
    print(f"{a} + {b} <= {c}")
elif a + c <= b:
    print("Pre trojuholník neplatí trojuholníková nerovnosť")
    print("a + c <= b")
    print(f"{a} + {c} <= {b}")
elif b + c <= a:
    print("Pre trojuholník neplatí trojuholníková nerovnosť")
    print("b + c <= a")
    print(f"{b} + {c} <= {a}")
else:
    alpha = math.acos((a**2 - b**2 - c**2) / (-2*b*c))

```

```
beta = math.acos((b**2 - a**2 - c**2) / (-2*a*c))
gamma = math.acos((c**2 - a**2 - b**2) / (-2*a*b))

va = c * math.sin(beta)
vb = a * math.sin(gamma)
vc = b * math.sin(alpha)

alpha = math.degrees(alpha)
beta = math.degrees(beta)
gamma = math.degrees(gamma)

print(f"\nStrany: a = {a}; b = {b}; c = {c}")
print(f"Uhly: alpha = {alpha}°; beta = {beta}°; gamma = {gamma}°")
print(f"Výšky: v(a) = {va}; v(b) = {vb}; v(c) = {vc}")
print(f"O = {a + b + c}")
print(f"S = {a * va * 0.5}")

print("Trojuholník je:", end=" ")
if a == b == c:
    print("Rovnostranný", end=" ")
elif a == b or b == c or c == a:
    print("Rovnoramenný", end=" ")
else:
    print("Rôznostranný", end=" ")

if alpha < 90 and beta < 90 and gamma > 90:
    print("Ostrouhlý")
elif alpha > 90 or beta > 90 or gamma > 90:
    print("Tupouhlý")
else:
    print("Pravouhlý")
```

### A.0.3 Cykly

#### 1. 100-krát napíš

```
veta = input("Musím napísať: ")
pocet = int(input("Toľkoto krát: "))
for i in range(pocet):
```

---

```
print(veta)
```

## 2. Hodnotenie

```
skore = int(input("Skóre: "))
for i in range(skore):
    print("*", end=" ")
print()
```

## 3. Pyramída

```
vyska = int(input("Výška pyramídy: "))
for riadok in range(vyska):
    medzery = vyska - riadok - 1
    hviezdy = 2 * riadok + 1
    print(" " * medzery + "*" * hviezdy)
```

## 4. Smaragd

```
vyska = int(input("Veľkosť: "))

if vyska < 3 or vyska % 2 != 1:
    print("Neviem vytvoriť taký smaragd")
else:
    vyska = (vyska // 2) + 1

    # Horná časť
    for riadok in range(vyska):
        medzery = vyska - riadok - 1
        hviezdy = 2 * riadok + 1
        print(" " * medzery + "*" * hviezdy)

    # Dolná časť
    for riadok in range(1, vyska):
        medzery = riadok
        hviezdy = 2 * (vyska - riadok) - 1
        print(" " * medzery + "*" * hviezdy)
```

## 5. Duté vnútro

```
vyska = int(input("Výška pyramídy: "))
for riadok in range(vyska):
    medzery = vyska - riadok - 1
    dute = 2 * riadok - 1

    print(" " * medzery, end="")
    if riadok == 0:
        print("*")
    elif riadok == vyska - 1:
        print("*" * (dute + 2))
    else:
        print("*" + " " * dute + "*")
```

## 6. Mriežka slov

```
n = int(input("Počet riadkov a stĺpcov: "))
slovo = input("Opakovať slovo: ")

for riadok in range(n):
    for stlpec in range(n):
        print(slovo, end=" ")
    print()
```

## 7. Rám

```
n = int(input("Počet riadkov a stĺpcov: "))
slovo = input("Opakovať slovo: ")
ram = len(slovo) * "#"

for riadok in range(n):
    for stlpec in range(n):
        if riadok == 0 or stlpec == 0 or riadok == n - 1 or stlpec == n - 1:
            print(ram, end=" ")
        else:
            print(slovo, end=" ")
    print()
```

---

## 8. Malá násobilka

```
for i in range(1, 11):
    for j in range(1, 11):
        print(f"{i * j:3d}", end=" ")
    print()
```

## 9. Sporenie

```
vkklad = float(input("Vklad v Eur: "))
sadzba = float(input("Úroková sadzba p.a. v \%: "))
urocenie = input("Typ úročenia (jednoduché / zložené): ")
ciel = float(input("Žiadaná suma v Eur: "))

sadzba /= 100
rok = 0
suma = vkklad

if urocenie == "jednoduché":
    urok = vkklad * sadzba
if urocenie == "zložené":
    sadzba += 1
    povodna_sadzba = sadzba

print(f"'Mesiac':10s} {'Suma':15s} {'Úrok':10s}")

while suma < ciel:
    if urocenie == "jednoduché":
        suma += urok
    elif urocenie == "zložené":
        urok = suma * (sadzba - 1)
        suma = vkklad * sadzba
        sadzba *= povodna_sadzba
    else:
        break
    rok += 1
    print(f"{rok:10d} {suma:15.2f} {urok:10.2f}")
```

## A.0.4 Náhodné čísla

### 1. Hádzanie kockou

```
import random
input("HOĎ")
kocka = random.randint(1, 6)

if kocka == 1:
    print("+-----+")
    print("|          |")
    print("|   #   |")
    print("|          |")
    print("+-----+")
elif kocka == 2:
    print("+-----+")
    print("| #      |")
    print("|          |")
    print("|      # |")
    print("+-----+")
elif kocka == 3:
    print("+-----+")
    print("| #      |")
    print("|   #   |")
    print("|      # |")
    print("+-----+")
elif kocka == 4:
    print("+-----+")
    print("| #   # |")
    print("|          |")
    print("| #   # |")
    print("+-----+")
elif kocka == 5:
    print("+-----+")
    print("| #   # |")
    print("|   #   |")
    print("| #   # |")
    print("+-----+")
elif kocka == 6:
```

---

```
print("+-----+")
print("| #   # |")
print("| #   # |")
print("| #   # |")
print("+-----+")
```

## 2. Hádaj číslo

```
import random
hadaj = random.randint(1, 100)
while True:
    tip = int(input("Hádaj číslo: "))
    if tip > hadaj:
        print("Veľa")
    elif tip < hadaj:
        print("Málo")
    else:
        print("Uhádol si")
        break
```

## 3. Opakovanie násobilky

```
import random
while True:
    x = random.randint(1, 10)
    y = random.randint(1, 10)
    print(f"Koľko je {x} x {y}?")
    vysledok = int(input("="))

    while vysledok != x * y:
        print("Nesprávne - hádaj znovu")
        vysledok = int(input("="))

    print("Správne - len tak ďalej")
    pokracuj = input("Chceš ďalší príklad? (a / n): ")
    if pokracuj == 'n':
        break
```



## A.0.5 Reťazce a zoznamy

### 1. Vymeň písmeno

```
text = input("Správa: ")
chyba = input("Za chybné písmeno: ")
nahrada = input("Vymeň: ")
upravene = ""
for pismeno in text:
    if pismeno == chyba:
        upravene += nahrada
    else:
        upravene += pismeno
print("\nOpravené!")
print(upravene)
```

### 2. Cenzúra

```
vstup = input("Správa: ")
prepis = input("Samohlásku nahraď: ")
vystup = ""
samohlasky = "aeiouyáéíóúý"
najdene = False

for i in range(len(vstup)):
    for j in range(len(samohlasky)):
        if vstup[i] == samohlasky[j]:
            vystup += prepis
            najdene = True
            break
    if not najdene:
        vystup += vstup[i]
    najdene = False

print("Cenzurované", vystup)
```

### 3. Počítanie slov

```
clanok = input("Článok: ")
```

```
pocet_znakov = 0
pocet_slov = 0
pocet_viet = 0
je_medzera = True

for znak in clanok:
    pocet_znakov += 1
    if znak == ".":
        pocet_viet += 1
    if znak.isspace():
        je_medzera = True
    elif je_medzera and not znak.isspace():
        pocet_slov += 1
        je_medzera = False

print(f"Znaky: {pocet_znakov}")
print(f"Slová: {pocet_slov}")
print(f"Vety: {pocet_viet}")
print(f"Normostany: {int(pocet_znakov / 1800)}")
```

#### 4. Najdlhšie slovo

```
prejav = input("Rečnícky prejav: ")
slovo = ""
najdlhsie = ""

for znak in prejav:
    if znak.isalpha():
        slovo += znak
    else:
        if len(slovo) > len(najdlhsie):
            najdlhsie = slovo
        slovo = ""

print(f"Najdlhšie slovo v ňom: {najdlhsie}")
```

#### 5. Frekvencia písmen

```
clanok = input("Článok: ")
```

```
abeceda = [0] * 26
pismena = 0

for pismeno in clanok:
    if pismeno.isalpha():
        pozicia = ord(pismeno.upper()) - ord("A")
        if pozicia >= 0 and pozicia <= 26:
            abeceda[pozicia] += 1
            pismena += 1

for i in range(len(abecedaRetazce a zoznamy - Riešenia)):
    pismeno = chr(ord("A") + i)
    vyskyt = 100 * (abeceda[i] / pismena)
    print(f"{pismeno}: {vyskyt:.2f}%")
```

## 6. Histogram

```
clanok = input("Článok: ")

STO_PERCENT = 100
abeceda = [0] * 26
pismena = 0

for pismeno in clanok:
    if pismeno.isalpha():
        pozicia = ord(pismeno.upper()) - ord("A")
        if pozicia >= 0 and pozicia <= 26:
            abeceda[pozicia] += 1
            pismena += 1

for i in range(len(abeceda)):
    pismeno = chr(ord("A") + i)
    vyskyt = int(STO_PERCENT * (abeceda[i] / pismena))
    print(f"{pismeno}: {'*' * vyskyt}")
```

## 7. Nákupný košík

```
nakup = []
while True:
    tovar = input("Čo kúpiť?: ")
```

---

```

    if tovar == "HOTOVO":
        break
    cena = float(input(f"Cena {tovar}?: "))
    nakup.append([tovar, cena])

riadok = "+" + 20 * "-" + "+" + 15 * "-" + "+" + 15 * "-" + "+"
print(riadok)
print(f"|{'Tovar':20s}|{'DPH':15s}|{'Cena s DPH':15s}|")

celkom = 0
for polozka in nakup:
    tovar = polozka[0]
    cena = polozka[1]
    celkom += cena
    print(riadok)
    print(f"|{tovar:20s}|{cena * 0.2:15.2f}|{cena:15.2f}|")

print(riadok)
print(f"|{'CELKOM':20s}|{celkom * 0.2:15.2f}|{celkom:15.2f}|")
print(riadok)

```

## 8. Akronym

```

veta = input("Slovné spojenie: ")
skratka = ""
je_medzera = True
for znak in veta:
    if znak.isspace():
        je_medzera = True
    elif je_medzera and znak.isalpha():
        je_medzera = False
        skratka += znak.upper()
print(f"Skratka: {skratka}")

```

## 9. Veľa opakovania

```

cesta = input("Cesta robota: ")
skratene = ""
smer = ""

```

```
n = 0
for krok in cesta:
    if krok.isalpha():
        if smer == "":
            smer = krok
            n = 1
        elif krok != smer:
            skratene += f"{n}{smer}"
            smer = krok
            n = 1
        else:
            n += 1
skratene += f"{n}{smer}"
print(f"Skomprimované: {skratene}")
```

## A.0.6 Súbory

### 1. Prepisovanie

```
nazov_suboru = input("Názov súboru")
subor = open(nazov_suboru, "r")
for riadok in subor:
    riadok = riadok.strip()
    ...
subor.close()
```

### 2. Turistika

```
nazov = input("Výškový profil trasy je v súbore: ")

ROVINA_KMH = 3.6
KROK_M = 10

predch_vyska = None
vzdialenost_m = 0
trvanie_min = 0

celkom_stupanie = 0
celkom_klesanie = 0
```

```
najvyssie = None
najnizsie = None

trasa = open(nazov, "r")

for miesto in trasa:
    nadmorska_vyska = int(miesto)
    vzdialenost_m += KROK_M

    # Ak neexistuje predošlá nadmorská výška, tak sme neprešli žiaden ú
    sek

    if predch_vyska != None:
        stupanie = nadmorska_vyska - predch_vyska

        # Zisti, či sme dosiahli rekordnú nadmorskú výšku a zaznamenaj si
        ju.
        if najvyssie == None or nadmorska_vyska > najvyssie:
            najvyssie = nadmorska_vyska
        elif najnizsie == None or nadmorska_vyska < najnizsie:
            najnizsie = nadmorska_vyska

        # Zobrazenie vzdialenosti medzi dvomi miestami zo svahu do roviny
        .

        # Pri stúpaní prejdeme za rovnaký čas akoby kratšiu vzdialenosť,
        preto
        # sa prepona zobrazí do dolnej odvesy a pri klesaní naopak
        if stupanie > 0:
            rovina_vzd = KROK_M ** 2 - stupanie ** 2
            celkom_stupanie += stupanie

        elif stupanie < 0:
            rovina_vzd = KROK_M ** 2 + stupanie ** 2
            celkom_klesanie += abs(stupanie)

        else:
            rovina_vzd = KROK_M

    # Čas na prejdienie medzi miestami v minútach
```

```
        trvanie_min += ((rovina_vzd / 1000) / ROVINA_KMH) * 60

    predch_vyska = nadmorska_vyska

trasa.close()

print(f"Trasa: {vzdialenost_m / 1000:.3f} km - "
      f"{int(trvanie_min // 60)} h {int(trvanie_min % 60)} min")
print(f"Stúpanie: {celkom_stupanie} m")
print(f"Klesanie: {celkom_klesanie} m")
print(f"Najnižšie miesto trasy: {najnizsie} m")
print(f"Najvyššie miesto trasy: {najvyssie} m")
```

### 3. Vedomostný kvíz

```
import random
nazov = input("Súbor s kvízovými otázkami: ")
kviz = open(nazov, "r")
otazky = []
skore = 0

# Ulož si pozície otázok v súbore
while True:
    riadok = kviz.readline()
    if not riadok:
        break
    if riadok.startswith("Otázka: "):
        znacka = kviz.tell() - len(riadok)
        otazky.append(znacka)
print("Kvízové otázky pripravené.")
print("Ideme na to!", end="\n\n")

while True:
    # Náhodne vyber otázku
    i = random.randint(0, len(otazky) - 1)
    znacka = otazky[i]
    kviz.seek(znacka)

    # Spýtaj sa otázku a navrhni možnosti
```

---

```

for riadok in kviz:
    riadok = riadok.rstrip()
    if riadok.startswith("Odpoveď: "):
        odpoved = riadok.lstrip("Odpoveď: ")
        break
    print(riadok.lstrip("Otázka: "))

# Hráčov tip
tip = input("Aká je správna odpoveď?: ")
if tip == odpoved:
    skore += 1
    print(f"Správne! Máš {skore} bodov.\n")
else:
    print("Nabudúce si to lepšie premysli. Skúsime niečo iné.\n")

kviz.close()

```

#### 4. Narodeniny

```

datum = input("Zobraz narodeniny pre mesiac v roku: ")
datum = datum.split(".")

NAZVY_MESIACOV = [
    "Január", "Február", "Marec", "Apríl", "Máj", "Jún", "Júl",
    "August", "September", "Október", "November", "December"
]
mesiac = int(datum[0])
rok = int(datum[1])

narodeniny = open("narodeniny.csv", "r")
print(f"\nNarodeniny: {NAZVY_MESIACOV[mesiac - 1]} {rok}")

for osoba in narodeniny:
    osoba = osoba.split(",")
    meno = osoba[0]
    datum = osoba[1].split(".")

    narodenie_den = int(datum[0])
    narodenie_mesiac = int(datum[1])

```



```
narodenie_rok = int(datum[2])

if narodenie_mesiac == mesiac:
    print(f"{narodenie_den}.{narodenie_mesiac} - {meno} - "
          f"{rok - narodenie_rok} rokov")

narodeniny.close()
```

## 5. Cestovné poriadky

```
odchod = input("Čas: ")
trvanie = input("Trvanie cesty vlakom: ")

odchod = odchod.split(":")
hod = int(odchod[0])
min = int(odchod[1])
odchod = [hod, min]

trvanie = trvanie.split(":")
hod = int(trvanie[0])
min = int(trvanie[1])
trvanie = [hod, min]

vlaky = []
autobusy = []
cp = open("cp.csv", "r")

for spoj in cp:
    spoj = spoj.split(",")
    doprava = spoj[0].strip()

    if doprava == "bus":
        autobusy.append([])

    for cas in spoj[1:]:
        cas = cas.split(":")
        hod = int(cas[0])
        min = int(cas[1])
```

---

```

        if doprava == "vlak":
            vlaky.append([hod, min])
        elif doprava == "bus":
            autobusy[-1].append([hod, min])

print("Najbližší spoj (vlak, autobus):")
nasiel = False

for vlak in vlaky:
    # Nájdi najbližší odchod vlaku
    if (vlak[0] * 60 + vlak[1]) >= (odchod[0] * 60 + odchod[1]):
        # Zisti, kedy prídeme odchod + trvanie = prichod
        min = (vlak[1] + trvanie[1]) % 60
        hod = ((vlak[0] + trvanie[0]) + ((vlak[1] + trvanie[1]) // 60)) %
            24
        prichod = [hod, min]

        for linka in autobusy:
            stanica = linka[0]

            # K tomu pozri autobusovú linku, ktorá odchádza zo stanice,
            do ktorej vlak ide
            if (stanica[0] * 60 + stanica[1]) >= (trvanie[0] * 60 +
                trvanie[1]):

                for autobus in linka[1:]:
                    # Prestup: Nájdi autobus, ktorý odchádza najskôr po
                    príchode vlaku
                    if (not nasiel and (autobus[0] * 60 + autobus[1]) > (
                        prichod[0] * 60 + prichod[1])):
                        print(f"{vlak[0]:02d}:{vlak[1]:02d} - "
                            f"{prichod[0]:02d}:{prichod[1]:02d}, "
                            f"{autobus[0]:02d}:{autobus[1]:02d} - ")
                        nasiel = True

cp.close()

```

## A.0.7 Funkcie

### 1. Vraký

```
import random
import math
MIERKA = 15
VRAK_X = random.randint(0, MIERKA)
VRAK_Y = random.randint(0, MIERKA)

def vzdialenost(x, y):
    return math.hypot(x - VRAK_X, y - VRAK_Y)

def nasiel(x, y):
    return x == VRAK_X and y == VRAK_Y

print("Sonar hlási potopený parník na dohľad!")
while True:
    suradnice = input("Tvoje súradnice?: ")
    suradnice = suradnice.split(",")
    x = int(suradnice[0])
    y = int(suradnice[1])
    if nasiel(x, y):
        print("Našiel si vrak. Dobrá práca!")
        break
    print(f"Od vraku si {vzdialenost(x, y):.3f} námorných míľ")
```

### 2. Cézarová šifra

```
def sifruj(sprava, kluc):
    sifra = ""
    A = ord("A")
    Z = ord("Z")
    ABECEDA = Z - A + 1
    sprava = sprava.upper()

    for i in range(len(sprava)):
        pismeno = sprava[i]
        k = kluc[i % len(kluc)]
```

---

```

        if A <= ord(pismeno) <= Z:
            poradie = ord(pismeno) - A
            posun = ord(k) - A
            poradie = (poradie + posun) % ABECEDA
            sifra += chr(poradie + A)
    return sifra

def desifruj(sifra, kluc):
    sprava = ""
    A = ord("A")
    Z = ord("Z")
    ABECEDA = Z - A + 1
    sifra = sifra.upper()

    for i in range(len(sifra)):
        pismeno = sifra[i]
        k = kluc[i % len(kluc)]
        if A <= ord(pismeno) <= Z:
            poradie = ord(pismeno) - A
            posun = ord(k) - A
            poradie = (poradie - posun) % ABECEDA
            sprava += chr(poradie + A)
    return sprava

retazec = input("Zadaj správu: ")
kluc = input("Vlož tajný kľúč: ")
akcia = input("Čo spraviť (šifruj / dešifruj): ")
s = ""
if akcia == "šifruj":
    print("Zašifrovaná správa: ", end="")
    s = sifruj(retazec, kluc)
elif akcia == "dešifruj":
    print("Dešifrovaná správa: ", end="")
    s = desifruj(retazec, kluc)
print(s)

```

### 3. Pascalov trojuholník

```
def pascalov_trojuholnik(n):
```

```
row = [1, 1]
medzery = n
pocet = 0

for i in range(n):
    pocet += 1
    medzery -= 1
    print(" " * medzery, end="")
    for cislo in row[:pocet]:
        print(cislo, end=" ")
    print()
    for j in range(pocet - 1, 0, -1):
        row[j] = row[j] + row[j - 1]
    row.append(1)

vyska = int(input("Zadajte výšku Pascalovho trojuholníka: "))
pascalov_trojuholnik(vyska)
```

#### 4. Pekný byt

```
import math

def priemer(zoznam):
    sucet = 0
    for prvok in zoznam:
        sucet += prvok
    return sucet / len(zoznam)

def modus(zoznam):
    nazvy = []
    vyskyty = []
    # Zisti koľkokrát sa čo vyskytuje
    for prvok in zoznam:
        index = -1
        for i in range(len(nazvy)):
            if prvok == nazvy[i]:
                index = i
        if index != -1:
            vyskyty[index] += 1
        else:
```

---

```

        nazvy.append(prvok)
        vyskyty.append(0)

# Pozri sa po najväčšom počte objavení sa a prehlás ho za modus
najviac = None
rekorder = -1
for i in range(len(vyskyty)):
    if najviac == None or vyskyty[i] > najviac:
        najviac = vyskyty[i]
        rekorder = i
return nazvy[rekorder]

def utried(zoznam):
    for i in range(len(zoznam) - 1):
        for j in range(len(zoznam) - i - 1):
            if zoznam[j] > zoznam[j + 1]:
                x = zoznam[j]
                zoznam[j] = zoznam[j + 1]
                zoznam[j + 1] = x

def median(zoznam):
    utried(zoznam)
    stred = (len(zoznam) + 1) // 2
    return zoznam[stred - 1]

def smerodajna_odchylka(zoznam):
    average = priemer(zoznam)
    sucet = 0
    for prvok in zoznam:
        sucet += (prvok - average) ** 2
    return math.sqrt(sucet / len(zoznam))

subor = input("Súbor s bytmi v lokalite: ")
ceny = []
vymery = []

byty = open(subor, "r")
for byt in byty:
    zaznam = byt.split(",")

```

```

ceny.append(int(zaznam[0]))
vymery.append(int(zaznam[1]))
byty.close()
# Pozri tiež modul "statistics" - https://docs.python.org/3/library/
statistics.html
print(f"{' ':25s}:{'Cena (eur)':15s}:{'Výmera(m^2)':15s}:")
print(f"{'Priemer':25s}:{priemer(ceny):15.2f}:{priemer(vymery):15.2f}:")
print(f"{'Medián':25s}:{median(ceny):15.2f}:{median(vymery):15.2f}:")
print(f"{'Modus':25s}:{modus(ceny):15.2f}:{modus(vymery):15.2f}:")
print(f"{'Smerodajná odchýlka':25s}:{smerodajna_odchylka(ceny):15.2f}:{
smerodajna_odchylka(vymery):15.2f}:")

```

## 5. Rímske čísla

```

def rimske_na_arabske(rimske):
    TABULKA = {"I": 1, "V": 5, "X": 10, "L": 50, "C": 100, "D": 500, "M":
1000}
    arabske = []
    vysledok = 0
    for symbol in rimske:
        arabske.append(TABULKA[symbol])
    i = 0
    while i < len(arabske):
        if i + 1 != len(arabske) and arabske[i] < arabske[i + 1]:
            vysledok += arabske[i + 1] - arabske[i]
            i += 2
        else:
            vysledok += arabske[i]
            i += 1
    return vysledok
cislo = input("Zadaj rímske číslo: ")
print(rimske_na_arabske(cislo))

```

## 6. Základný tvar zlomku

```

def nsd(a, b):
    # Najväčší spoločný deliteľ
    # alebo: math.gcd(a, b)
    while b > 0:

```

---

```

        a, b = b, a \% b
    return a

def nsn(a, b):
    # Najmenší spoločný násobok
    return a * b // nsd(a, b)

def zakladny_tvar(zlomok):
    delitel = nsd(zlomok[0], zlomok[1])
    return [
        zlomok[0] // delitel,
        zlomok[1] // delitel
    ]

def vytvor_zlomok(retazec):
    # alebo: map(int, retazec.split("/"))
    zlomok = retazec.split("/")
    for i in range(len(zlomok)):
        zlomok[i] = int(zlomok[i])
    return zlomok

def nasobit(x, y):
    citatel = x[0] * y[0]
    menovatel = x[1] * y[1]
    zlomok = [citatel, menovatel]
    return zakladny_tvar(zlomok)

def delit(x, y):
    citatel = x[0] * y[1]
    menovatel = x[1] * y[0]
    zlomok = [citatel, menovatel]
    return zakladny_tvar(zlomok)

def scitat(x, y):
    menovatel = nsn(x[1], y[1])
    x_citatel = x[0] * (menovatel // x[1])
    y_citatel = y[0] * (menovatel // y[1])
    citatel = x_citatel + y_citatel
    zlomok = [citatel, menovatel]

```



```

    return zakladny_tvar(zlomok)

def odcitat(x, y):
    menovatel = nsn(x[1], y[1])
    x_citatel = x[0] * (menovatel // x[1])
    y_citatel = y[0] * (menovatel // y[1])
    citatel = x_citatel - y_citatel
    zlomok = [citatel, menovatel]
    return zakladny_tvar(zlomok)

def vypis(zlomok):
    return f"{zlomok[0]}/{zlomok[1]}"

print("Kalkulačka zlomkov")
a = input("a = ")
b = input("b = ")
akcia = input("Vypočítaj (+, -, *, /): ")

a = zakladny_tvar(vytvor_zlomok(a))
b = zakladny_tvar(vytvor_zlomok(b))

print("\nVýsledok:")
if akcia == '+':
    print(f"{vypis(a)} + {vypis(b)} = {vypis(scitat(a, b))}")
elif akcia == '-':
    print(f"{vypis(a)} - {vypis(b)} = {vypis(odcitat(a, b))}")
elif akcia == '*':
    print(f"{vypis(a)} * {vypis(b)} = {vypis(nasobit(a, b))}")
elif akcia == '/':
    print(f"{vypis(a)} / {vypis(b)} = {vypis(delit(a, b))}")

```

# Príloha B: Zaradenie úloh do zbierky

úloha kategória	1.	2.	3.	4.	5.	6.	7.	8.	9.	10.	11.
upevňovanie učiva		×	×		×	×	×				
aplikácia mimo odbor		×	×	×	×	×	×	×	×	×	×
aplikácia vo vnútri odboru											
opakovanie a systematizácia			×			×	×	×	×	×	
aktualizačné úlohy				×							
prípravné úlohy	×										
osvojenie pojmu a postupu	×	×		×							
motivačné úlohy	×	×		×	×						×
propedeutické úlohy	×			×							
nižšie konvergentné procesy	×	×	×	×			×	×	×	×	
vyššie konvergentné procesy					×	×					×
počet slov	56	49	47	41	67	70	43	48	42	54	64
počet viet	6	4	5	4	7	7	4	5	4	6	6
náročné slová	16	13	15	10	21	22	11	21	9	24	26
fog index	15,16	15,51	16,53	13,86	16,37	16,57	14,53	21,34	12,77	21,38	20,52

Tabuľka B.1: Premenné

úloha kategória	1.	2.	3.	4.	5.	6.	7.	8.	9.
upevňovanie učiva			×	×					
aplikácia mimo odbor		×	×	×	×		×	×	×
aplikácia vo vnútri odboru	×					×			
opakovanie a systematizácia				×	×	×	×	×	×
aktualizačné úlohy	×	×			×			×	
prípravné úlohy	×								
osvojenie pojmu a postupu	×	×		×					
motivačné úlohy			×	×			×	×	×
propedeutické úlohy	×								
nižšie konvergentné procesy	×	×			×	×		×	
vyššie konvergentné procesy				×			×		×
hodnotiace myslenie			×						
počet slov	33	30	27	82	42	36	90	54	85
počet viet	4	4	3	7	5	4	6	4	7
náročné slová	10	6	9	29	24	13	27	17	30
fog index	15,42	11,00	16,93	18,83	26,22	18,04	18,00	17,99	18,97

Tabuľka B.2: Podmienky

úloha kategória	1.	2.	3.	4.	5.	6.	7.	8.	9.
upevňovanie učiva			×	×		×	×		
aplikácia mimo odbor									×
aplikácia vo vnútri odboru									
opakovanie a systematizácia				×	×		×	×	×
aktualizačné úlohy						×			
prípravné úlohy	×	×							
osvojenie pojmu a postupu	×	×	×			×		×	
motivačné úlohy			×						×
propedeutické úlohy	×	×				×			
nižšie konvergentné procesy	×	×		×	×	×	×	×	
vyššie konvergentné procesy			×						×
počet slov	34	26	72	76	28	48	43	26	47
počet viet	3	3	6	7	3	3	5	2	4
náročné slová	12	14	30	27	14	17	12	11	18
fog index	18,65	25,01	21,47	18,55	23,73	20,57	14,60	22,12	20,02

Tabuľka B.3: Cykly

úloha kategória	1.	2.	3.
upevňovanie učiva		×	×
aplikácia mimo odbor		×	
aplikácia vo vnútri odboru	×		×
opakovanie a systematizácia		×	×
aktualizačné úlohy	×	×	×
prípravné úlohy			
osvojenie pojmu a postupu	×	×	
motivačné úlohy	×		
propedeutické úlohy	×		
nižšie konvergentné procesy	×		×
vyššie konvergentné procesy		×	
hodnotiace myslenie			
divergentné myslenie			
počet slov	46	68	33
počet viet	5	7	4
náročné slová	18	19	10
fog index	19,33	15,06	15,42

Tabuľka B.4: Náhodné čísla

úloha kategória	1.	2.	3.	4.	5.	6.	7.	8.	9.
upevňovanie učiva		×	×	×	×		×		
aplikácia mimo odbor		×	×				×		
aplikácia vo vnútri odboru	×			×	×	×			×
opakovanie a systematizácia						×	×	×	×
aktualizačné úlohy	×	×		×					
prípravné úlohy									
osvojenie pojmu a postupu	×	×		×	×				
motivačné úlohy	×	×		×	×				×
propedeutické úlohy									
nižšie konvergentné procesy	×							×	
vyššie konvergentné procesy		×	×	×	×	×	×		×
počet slov	42	21	43	36	40	31	52	27	63
počet viet	4	2	3	4	3	3	4	3	6
náročné slová	12	5	14	15	12	12	15	10	17
fog index	15,63	13,72	18,76	20,27	17,33	19,62	16,74	18,41	14,99

Tabuľka B.5: Reťazce a zoznamy

úloha kategória	1.	2.	3.	4.	5.
upevňovanie učiva		×	×	×	×
aplikácia mimo odbor		×	×		
aplikácia vo vnútri odboru				×	
opakovanie a systematizácia	×		×	×	
aktualizačné úlohy	×	×		×	
prípravné úlohy					
osvojenie pojmu a postupu	×	×		×	
motivačné úlohy		×	×	×	
propedeutické úlohy	×				
nižšie konvergentné procesy	×				
vyššie konvergentné procesy		×		×	×
hodnotiace myslenie					
divergentné myslenie			×		
počet slov	44	63	66	37	51
počet viet	4	5	7	3	5
náročné slová	15	26	8	11	18
fog index	18,04	21,55	20,74	16,83	18,20

Tabuľka B.6: Súborny

úloha kategória	1.	2.	3.	4.	5.	6.	7.	8.
upevňovanie učiva		×	×	×	×			
aplikácia mimo odbor	×		×	×	×	×		×
aplikácia vo vnútri odboru		×					×	
opakovanie a systematizácia				×	×	×	×	×
aktualizačné úlohy	×	×			×			
prípravné úlohy	×							
osvojenie pojmu a postupu	×	×						
motivačné úlohy	×	×		×	×	×		×
propedeutické úlohy	×							
nižšie konvergentné procesy	×							
vyššie konvergentné procesy		×	×					
hodnotiace myslenie								
divergentné myslenie				×	×	×	×	×
počet slov	48	102	31	49	51	28	72	65
počet viet	4	8	3	4	5	3	7	5
náročné slová	20	26	9	17	17	10	29	21
fog index	21,23	15,30	15,75	18,78	17,41	18,02	20,23	18,12

Tabuľka B.7: Funkcie

# Príloha C: Výpočet skóre čitateľnosti

Na výpočet počtu slov, počet viet a fog indexu sme použili vlastný program v jazyku Python:

```
import pyphen
from nltk.tokenize import sent_tokenize, RegexpTokenizer

word_tokenizer = RegexpTokenizer(r"\w+")
dic = pyphen.Pyphen(lang="sk_SK")

def count_syllables(word: str) -> int:
    return len(dic.inserted(word).split("-"))

def fog_index(text: str) -> float:
    sentences = sent_tokenize(text)
    words = word_tokenizer.tokenize(text)
    long_words = [w for w in words if count_syllables(w) > 2]

    print(f"Pôčet slov: {len(words)}")
    print(f"Pôčet viet: {len(sentences)}")
    print(f"Náročné slová: {len(long_words)}")

    psv = len(words) / len(sentences)
    ds = (len(long_words) / len(words)) * 100

    score = 0.4 * (psv + ds)
    return score

if __name__ == "__main__":
    text = input("> ")
    print(f"Fog index: {fog_index(text):.2f}")
```