INFORMATIKA

K možnosti využití programování při žákovském řešení vybraných matematických úloh MO

LADISLAV PERK

Přírodovědecká fakulta UJEP, Ústí nad Labem

V pedagogické realitě se lze setkat s žáky, kteří rádi řeší úlohy Matematické olympiády (dále jen MO). Tito žáci se mohou dostat do situace, kdy jim nalezení řešení matematickými prostředky může činit obtíže a nalezení správného řešení úlohy je pro ně buď velmi obtížné, anebo prakticky nemožné.

Mezi těmito žáky se mohou nacházet tací, kteří mají kladný vztah k programování a sami programují. Je však žádoucí, aby měli alespoň základní zkušenosti s programováním. Pokud je matematická úloha řešitelná pomocí programování, pak napsáním správně fungujícího programu mohou žáci získat správné výsledky úlohy: vytvořený program jim správné výsledky nalezne a vypíše. Pokud se žáci rozhodnou nalézt řešení úlohy pomocí programování, pak musí být schopni napsat funkční program, který prohledá celý stavový prostor. Prohledáním stavového prostoru budeme rozumět posouzení všech variant řešení z množiny konečného počtu potenciálních výsledků, zda splňují všechny podmínky formulované v zadání úlohy. Pokud daná varianta všechny tyto podmínky splňuje, pak může být programem vypsána jako řešení úlohy.

Lze se domnívat, že pro některé žáky může být takový způsob hledání řešení úloh atraktivní. Zároveň se při tomto způsobu řešení úloh posilují kompetenci k řešení problémů, digitální kompetenci a mezipředmětové vztahy mezi matematikou a informatikou. V současnosti je takový způsob řešení úloh přínosný zejména v kontextu nového pojetí informatiky [1].

Zároveň je důležité sdělit, že představené zdrojové kódy v tomto článku nejsou z didaktického hlediska optimalizovány. Cílem tohoto článku je umožnit pochopení podstaty zdrojových kódů všem čtenářům – tedy i těm, kteří s programováním nemají žádné zkušenosti, či mají zkušenosti s programováním na velmi omezené úrovni. Čtenář po úspěšné implementaci a pochopení principu zdrojových kódů může – samozřejmě – tyto zdrojové kódy optimalizovat.

Žákovské hledání řešení vybraných matematických úloh MO pomocí programování: motivace

V zadání matematických úloh MO se vyskytují úlohy, jejichž řešení lze nalézt pomocí systematického experimentování, resp. použití hrubé síly [3]. Pro tuto skupinu úloh je vhodné využít počítač jako prostředek pro jejich vyřešení.

Systematické experimentování lze chápat jako metodický a strukturovaný přístup k hledání řešení matematických problémů. Tato metoda je založena na systematickém provádění pokusů, kdy se vyčerpávají jednotlivé z množiny všech potenciálních výsledků [2, 4, 5]. Použití hrubé síly budeme chápat jako využití systematického experimentování při řešení úloh za pomoci počítače [2]. Prohledávání stavového prostoru či průchod stavovým prostorem budeme – v kontextu tohoto článku – chápat jako systematické vyčíslování řídících proměnných a následné posuzování, zda výsledky vyčíslené z řídících proměnných splňují všechny podmínky zadání dané úlohy

Pokud se žáci rozhodnou řešit matematické úlohy MO pomocí prohledávání stavového prostoru, mohou využít řešení:

(A) „ručně“;
(B) pomocí tabulkového procesoru;
(C) pomocí programování.

Pokud se žáci rozhodnou řešit úlohy pomocí „ručního“ řešení (A), prohledávají stavový prostor bez výpočetní techniky, například skrze psaní na papír. Mohou si sestavit vhodnou tabulku a do ní dosazují.

Ukázkou takové řešení může být zde prezentovaná úloha 2, kdy proměnné a a b systematicky nabývají číselných hodnot od 1 do 9 a posuzuje se splnění všech podmínek zadání úlohy. Je vhodné, aby si žáci pro takové systematické prohledávání stavového prostoru vytvořili vhodnou krokovací tabulku (složenou z proměnných a a b), např. tab. 1. Stavový prostor tvoří $9 \cdot 9 = 81$ posouzení.

Výhodou tohoto přístupu je, že žáci „vidí“ proces systematického vy-

číslování a vyhodnocování podmínek. Naopak nevýhodou je, že pro větší stavový prostor může být pro žáky unavující a odrazující.

Tabulka 1 Ukázka sestavení krokovací tabulky pro systematické vyčíslování a vyhodnocování pomocí „ručního řešení úlohy 2“

a	b	$\overline{ab}$	$\overline{ba}$	$\overline{ab} - \overline{ba}$	Platí $\overline{ab} - \overline{ba} = 63$?
1	1	11	11	0	ne
1	2	12	21	-9	ne
1	3	13	31	-18	ne
⋮	⋮	⋮	⋮	⋮	⋮
⋯	⋯	⋯	⋯	63	ano
⋮	⋮	⋮	⋮	⋮	⋮
9	8	98	89	9	ne
9	9	99	99	0	ne

Pokud žáci zvolí řešení pomocí tabulkového procesoru (B), získávají možnost posouzení většího posouzení číselných hodnot. Přesto jsou omezeni maximálním počtem řádků či sloupců tabulkové procesoru. Např. pokud využijí MS Excel, pak soubor s extenzí .xls bude mít maximálně 65 536 řádků, soubor s extenzí .xlsx pak 1 048 576 řádků. Pokud řešení úlohy vyžaduje větší počet posouzení než tyto maximální počty tabulkových procesorů, jsou tabulkové procesory již nepoužitelné. Ukázkou takového řešení může být úloha 3, ve které je zapotřebí prozkoumat 8 999 variant. Takový počet je vhodný pro využití některého z tabulkových procesorů.

Pokud žáci zvolí možnost využití programování při řešení úloh (C), mají možnost prohledávání výrazně větších stavových prostorů. Ukázkou takového řešení může být úloha 1, kde je třeba posoudit $899 \cdot 899 \cdot 899 = 72\,572\,699$ variant. Využití tabulkového procesu je nevhodné, proto je nutné tuto úlohu řešit pomocí programování.

Vybrané úlohy MO řešitelné žákovským programováním

Nyní bude představeno celkem sedm vybraných matematických úloh MO, jejichž řešení lze nalézt napsáním funkčního programu, který požadovaná řešení vypíše. V tomto článku jsou programy napsány v programovacím jazyce Python.

Následující úloha je z hlediska obtížnosti jednou z nejjednodušších úloh nejen v rámci tohoto článku, ale také matematických úloh prezentovaných v MO.

Úloha 1 (viz [6])

Najděte všechny trojice trojmístných přirozených čísel a, b, c, pro která platí $b^2 = a \cdot c$, $b = a + 34$.

Autorské řešení. V úloze se hledají všechny trojice trojmístných přirozených čísel $\overline{abc}$, které musí splňovat požadované vlastnosti. Protože se ale v celém řešení nepracuje s dílčími číslicemi hledaných trojmístných čísel, není proto nutné formulovat každé hledané trojmístné číslo pomocí tří proměnných reprezentující jednotlivé číslice. Pro účely vyčíslení trojmístných čísel a, b a c budou uplatněny tři vzájemně vnořené cykly s pevným počtem opakování – for cykly s řídícími proměnnými `a`, `b`, a `c` (ř. 1–3). Tyto řídící proměnné budou nabývat celočíselných hodnot od 100 do 999.

V této fázi řešení je možné přistoupit k posouzení splnění rovností levých a pravých stran u obou rovnic v zadání úlohy. Pro žádnou ze stran není nutné zavádět pomocné proměnné, program je dostatečně přehledný. Toto posouzení lze zrealizovat zápisem obou rovnic jako dvou dílčích podmínek v konjunktivním tvaru v jednom podmíněném příkazu (ř. 4). V případě kladného vyhodnocení obou podmínek lze vypsat číselné hodnoty proměnných `a`, `b`, a `c`, které jsou hledanými řešeními úlohy (ř. 5–8).

```
1 for a in range(100, 1000):
2     for b in range(100, 1000):
3         for c in range(100, 1000):
4             if b * b == a * c and b == a + 34:
5                 print("a = ", a)
6                 print("b = ", b)
7                 print("c = ", c)
8                 print()
```

Program vypíše jako řešení dvě trojčíslí (a, b, c): (289, 323, 361) a (578, 612, 648). Dosazením do obou rovnic všech tří proměnných lze ověřit správnost vypsaných výsledků programem. Stejně tak lze ověřit správnost těchto výsledků v matematickém řešení úlohy na webových stránkách MO.

Následující úloha je ukázkou práce s jednotlivými číslicemi dvojmístných čísel, kdy dvojmístné číslo je nutné zformulovat pomocí dekadického zápisu.

Úloha 2 (viz [7])

Petr řekl Pavlovi: „Napiš dvojmístné přirozené číslo, které má tu vlastnost, že když od něj odečteš totéž dvojmístné přirozené číslo akorát napsané obráceně, dostaneš rozdíl 63.“ Které číslo mohl Pavel napsat? Určete všechny možnosti.

Autorské řešení. V úloze se pracuje s jednotlivými číslicemi dvojmístných čísel, proto je nutné zavést vhodně pojmenovanou proměnnou pro každý řád dvojmístného čísla, např. a a b (např. a bude reprezentovat desítky a b jednotky). Protože obě proměnné reprezentují číslice dvojmístných čísel, jejich přípustnými hodnotami jsou přirozená čísla od 0 do 9. Z důvodu zachování dvojmístnosti hledaného čísla a čísla s opačně zapsanými číslicemi vůči tomuto číslu mohou obě proměnné nabývat hodnot od 1 do 9. Pro vyčíslení proměnných a a b je nutné uplatnit dva vzájemně vnořené cykly s pevným počtem opakování – for cykly s řídícími proměnnými `a` a `b` (ř. 1 a 2).

Hledané dvojmístné $\overline{ab}$ a jeho opačně zapsané přirozené číslo $\overline{ba}$ je nutné v programu zformulovat pomocí příslušného dekadického zápisu pomocí proměnných `a` a `b` (ř. 3 a 4). Po vyčíslení obou dvojmístných čísel v zavedených pomocných proměnných `ab` a `ba` lze přistoupit k posouzení splnění velikosti rozdílu obou dvojmístných čísel (ř. 6). V případě kladného vyhodnocení této podmínky je možné vypsat číselnou hodnotu hledaného dvojmístného čísla (ř. 7), informativně pak hodnotu opačně zapsaného čísla (ř. 8).

```
1 for a in range(1, 10):
2     for b in range(1, 10):
3         ab = a * 10 + b
4         ba = b * 10 + a
5 
6         if ab - ba == 63:
7             print("1. cislo: ", ab)
8             print("2. cislo: ", ba)
9             print()
```

Program vypíše dvě dvojice přirozených čísel (ab, ba): (81, 18) a (92, 29). Jednoduchým výpočtem lze ověřit správnost obou vypsaných výsledků. Stejně tak lze ověřit správnost těchto výsledků v matematickém řešení úlohy na webových stránkách MO.

Následující úloha je ukázkou práce s palindromy a dělitelností přirozených čísel.

Úloha 3 (viz [8])

Pětimístným palindromem myslíme takové pětimístné číslo, které má na místě jednotek stejnou číslici jako na místě desetitisíců a na místě desítek stejnou číslici jako na místě tisíců. Najděte nejmenší pětimístný palindrom dělitelný 36.

Autorské řešení. V úloze se hledá nejmenší z pětimístných čísel, které má být palindromem a zároveň dělitelný číslem 36. Protože se v rámci řešení

pracuje s jednotlivými číslicemi hledaných pětimístných čísel, je nutné pro každou z pěti číslic zavést vhodnou pomocnou, např. a, b, c, d a e. Hledané pětimístné číslo lze zapsat jako $\overline{abcde}$, jeho opačně zapsané číslo pak $\overline{edcba}$.

Pro účely vyčíslení proměnných a až e v programu je nutné uplatnit pět vzájemně vnořených cyklů s pevným počtem opakování – for cyklů s řídícími proměnnými `a`, `b`, `c`, `d` a `e` (ř. 1–5). Všechny tyto proměnné představují číslice 0 až 9, avšak z důvodu zachování pětimístnosti čísel $\overline{abcde}$ a $\overline{edcba}$ nesmí proměnné nabývat `a` a `e` hodnoty 0. Pro účely zvýšení přehlednosti programu je vhodné zformulovat čísla $\overline{abcde}$ a $\overline{edcba}$ pomocí vhodných pomocných proměnných v dekadickém zápisu (ř. 6 a 7).

Nyní lze přistoupit k posouzení, zda pětimístné číslo $\overline{abcde}$ je palindromem (tj. roven opačně zaspanému číslu $\overline{edcba}$) a zároveň beze zbytku dělitelné číslem 36. Tyto dvě podmínky zapíšeme v jednom podmíněném příkazu v konjunktivním tvaru (ř. 9). V případě kladného posouzení lze vypsat číselnou hodnotu čísla $\overline{abcde}$ (ř. 10). Jakmile je tato hodnota vypsána, pak lze hledání dalších hodnotu ukončit pomocí `exit()`.

```
1  for a in range(1, 10):
2      for b in range(0, 10):
3          for c in range(0, 10):
4              for d in range(0, 10):
5                  for e in range(1, 10):
6                      abcde = a * 10000 + b * 1000 + c * 100 + d * 10
                           + e
7                      edcba = e * 10000 + d * 1000 + c * 100 + b * 10
                           + a
8
9                      if abcde == edcba and abcde % 36 == 0:
10                         print(abcde)
11                         exit()
```

Program vypíše jako řešení pětimístné číslo 21 312. Správnost vypsaného čísla lze ověřit v matematickém řešení úlohy na webových stránkách MO.

Následující úloha je zaměřena na práci s jednotlivými číslicemi osmimístných čísel a stanovení jejich počtu. Zajímavostí této úlohy je, že v průběhu řešení není zapotřebí formulovat hledaná osmimístná čísla (např. pomocí dekadického zápisu).

Úloha 4 (viz [9])

V osmimístném čísle je každá jeho číslice (kromě poslední) větší než číslice následující. Kolik je všech takových čísel?

Autorské řešení. V úloze se hledají všechna taková osmimístná čísla, u kterých každá číslice (kromě poslední) je větší než její následující číslice na

nižším číselném řádu (samozřejmě kromě poslední číslice). Pokud zapíšeme hledané osmimístné číslo jako $\overline{abcdefgh}$, pak platí:

$$a > b > c > \ldots > g.$$

Pro účely vyčíslení dílčích číselných řádů osmimístného čísla je nutné uplatnit osm vzájemně vnořených cyklů s pevným počtem opakování – for cyklů – s řídícími proměnnými `a` až `h` (ř. 3–10). Protože každá z proměnných představuje jednotlivé číslice, budou proměnné nabývat celočíselných hodnot od 0 do 9. Z důvodu zachování osmimístnosti hledaných osmimístných čísel však nesmí být číslice na nejvyšším číselném řádu nulová, tj. proměnná `a` nesmí být rovna 0.

Nyní je zapotřebí přistoupit k posouzení, zda každá číslice je větší než následující číslice na nižším číselném řádu (kromě poslední číslice). To se zrealizuje pomocí dílčího posouzení $a > b$ až $f > g$ v konjunktivním tvaru v podmíněném příkazu (ř. 11).

Protože v rámci řešení nepracujeme s hledanými osmimístnými čísly, ale pouze jejich číslicemi, není tedy zapotřebí hledaná osmiciferná čísla formulovat pomocí příslušných dekadických zápisů. V úloze se však hledá počet takových čísel. Proto je nutné zavést vhodně pojmenovanou pomocnou proměnnou a před započetím hledání počtu vynulovat (ř. 1). V případě kladného posouzení všech podmínek z předchozího odstavce je nutné inkrementovat proměnnou registrující tento počet (ř. 12). Po ukončení prohledávání stavového prostoru je možné vypsat hledaný počet těchto čísel (ř. 14).

```
1  pocet = 0
2
3  for a in range(1, 10):
4      for b in range(0, 10):
5          for c in range(0, 10):
6              for d in range(0, 10):
7                  for e in range(0, 10):
8                      for f in range(0, 10):
9                          for g in range(0, 10):
10                             for h in range(0, 10):
11                                 if a > b and b > c and  c> d and d
                                       > e and e > f and f > g and g
                                       > h:
12                                     pocet = pocet + 1
13
14 print("Nalezeny pocet: ", pocet)
```

Program vypíše číslo 45 jako hledaný počet. Správnost vypsaného čísla lze ověřit v matematickém řešení úlohy na webových stránkách MO.

V následující úloze se pracuje s dělitelností přirozených čísel.

Úloha 5 (viz [10])

Pinocchio tvrdí, že číslo dne v datu jeho narození lze beze zbytku dělit třemi, čtyřmi, pěti a šesti. Tři z těchto čtyř informací jsou pravdivé, jedna je nepravdivá. Kolikátý den v měsíci může mít Pinocchio narozeniny? Určete všechny možnosti.

Autorské řešení. V úloze se hledá pořadí dne či dnů v nekonkrétním měsíci, které mají splňovat podmínky dělitelnosti. Proto zavedeme vhodnou pomocnou proměnnou, např. d. Ze zadání úlohy musí platit podmínky dělitelnosti: daný den či dny d lze beze zbytku dělit právě třemi čísly ze čtyř čísel, kterými jsou čísla 3, 4, 5 a 6. Každý den či dny d vypsané jako řešení budou tedy dělitelné beze zbytku právě jednou ze čtyř trojicí: (4, 5, 6), (3, 5, 6), (3, 4, 6) a (3, 4, 5). Zároveň platí, že se jedná o dny v nekonkrétním měsíci, budeme proto uvažovat, že proměnná d může nabývat číselných hodnot od 1 do 31.

Pro účely vyčíslení dní v daném měsíci uplatníme jeden cyklus s pevným počtem opakování – for cyklus – s řídící proměnnou `d` (ř. 1). Tato proměnná bude nabývat číselných hodnot od 1 do 31.

Protože má nastat právě jedna trojice číslic ze čtyř číselných trojic, jedná se tedy o disjunktní podmínky. Posouzení dělitelnosti každou trojicí bude provedeno zvlášť: dělitelnost čísly 4, 5 a 6 jako dílčí podmínky v konjunktivním tvaru zápisu v podmíněném příkazu (ř. 3), dělitelnost čísly 3, 5 a 6 jako dílčí podmínky v konjunktivním tvaru zápisu v podmíněném příkazu (ř. 6), analogicky stejně pak pro zbylé trojice (ř. 9 a 12).

S ohledem na skutečnost, že pro den či dny d, které má či mají být řešením, bude kladně posouzena právě jedna ze čtyř posouzení (ř. 3, 6, 9, 12), musí každá z těchto podmínek umožňovat výpis d jako nalezeného řešení (ř. 4, 7, 10, 13).

```
1  for d in range(1, 32):
2
3      if d%4==0 and d%5==0 and d%6==0:
4          print(d)
5
6      if d%3==0 and d%5==0 and d%6==0:
7          print(d)
8
9      if d%3==0 and d%4==0 and d%6==0:
10         print(d)
11
12     if d%3==0 and d%4==0 and d%5==0:
13         print(d)
```

Program vypíše jako řešení tři dvojmístná čísla: 12, 24 a 30. Jednoduchými výpočty lze ověřit, že vypsaná řešení odpovídají všem podmínkám v zadání úlohy. Stejně tak lze ověřit správnost těchto výsledků v matematickém řešení úlohy na webových stránkách MO.

Následující úloha je unikátní tím, že v příkladu, který se skládá z šesti čísel a pěti znamének +, se systematicky mění znaménka + a −; vyčísluje se výsledek tohoto příkladu a vyhodnocuje dělitelnost výsledku příkladu.

Úloha 6 (viz [11])

V následujícím příkladu je pětkrát použito znaménko + a výsledek je násobkem tří:

$$9 + 8 + 7 + 6 + 5 + 4 = 39.$$

Změňte dvě ze znamének + na znaménko − tak, aby výsledek nového příkladu byl opět násobkem tří. Najděte všechny možnosti.

Autorské řešení. V úloze se mění znaménka + a − tak, aby po vyčíslení výsledek příkladu v zadání úlohy je beze zbytku dělitelný číslem 3.

Příklad je složen z celkem pěti znamének +, z nichž právě dvě znaménka + mají být zaměněny za znaménko −. Proto je nutné v programu zformulovat tuto systematickou záměnu. Jako jedno z možných řešení je, že zavedeme pět proměnných, např. $z_1, z_2, \ldots, z_5$, ve kterých budou systematicky měněny určité vhodné číselné hodnoty, např. 0 nebo 1. Zároveň zavedeme proměnnou reprezentující průběžný výsledek příkladu, např. `vysledek` kdy budou podle předchozích hodnot 0 nebo 1 vkládány znaménka + nebo − a podle toho daná čísla přičítána či odečítána od proměnné `vysledek`. Podle hodnoty 0 nebo 1 bude buď přičteno číslo za vkládaným znaménkem (při čísle 0) nebo odečteno číslo za vkládaným znaménkem (při čísle 1). Např. bude posuzována číselná pětice $(z_1, z_2, z_3, z_4, z_5) = (1, 1, 0, 0, 1)$, pak příklad nabude tvaru $9 - 8 - 7 + 6 + 5 - 4 = 1$. Výsledek 1 nemůže být řešením úlohy, neboť není násobkem čísla 3 a zároveň nebyly změněny právě dvě znaménka, ale tři. Naopak, pokud bude platit posuzována například číselná pětice $(z_1, z_2, z_3, z_4, z_5) = (0, 1, 0, 1, 0)$, pak příklad nabude tvaru $9 + 8 - 7 + 6 - 5 + 4 = 15$, což je akceptovaným výsledkem, neboť 15 je násobkem čísla 3 a byly změněny právě dvě znaménka.

Nyní lze přistoupit ke tvorbě programu. Pro účely vyčíslení proměnných z_1 až z_5 zavedeme pět vzájemně vnořených cyklů s pevným počtem opakování – for cyklů – s řídícími proměnnými `z1` až `z5`, které budou nabývat číselných hodnot 0 nebo 1 (ř. 1–5). Posléze proměnnou `vysledek` pro postupné sčítání či odčítání čísel. Tuto proměnnou lze nastavit na hodnotu

prvního čísla, tj. 9 (ř. 7). Zároveň je nutné zavést proměnnou, která bude reprezentovat počet změněných znamének, např. `pocet` (ř. 6).

Podle hodnoty `z1` se přičte nebo odečte číslo 8: v případě, že hodnota `z1` bude rovna 0 (ř. 9), pak se k proměnné `vysledek` přičte hodnota 8 (ř. 8), v opačném případě – kdy je hodnota `z1` rovna 1 (ř. 11) – se odečte číslo 8 (ř. 12). Analogicky postupujeme pro přičtení či odečtení čísla 7 skrze poměnnou `z2` (ř. 16–19), pro přičtení či odečtení čísla 6 skrze proměnnou `z3` (ř. 23–26) apod. Pro sledování počtu změněných znamének z + na − se inkrementuje proměnná `pocet` u každého odečtení čísla (ř. 13, 20, 27, 34, 41).

V případě, že hodnota proměnné `pocet` je rovna 2 (tj. byly změněny právě dvě znaménka z + na −) a zároveň hodnota proměnné `vysledek` je beze zbyku dělitelná číslem 3 (ř. 44), pak lze vypsat číselné hodnoty z_1 až z_5 (ř. 45–49) a příslušnou hodnotu proměnné `vysledek` (ř. 50), neboť tato čísla tvoří řešení úlohy.

```
1  for z1 in range(0, 2):
2      for z2 in range(0, 2):
3          for z3 in range(0, 2):
4              for z4 in range(0, 2):
5                  for z5 in range(0, 2):
6                      pocet = 0
7                      vysledek = 9
8
9                      if z1==0:
10                         vysledek = vysledek + 8
11                     else:
12                         vysledek = vysledek - 8
13                         pocet = pocet + 1
14
15
16                     if z2==0:
17                         vysledek = vysledek + 7
18                     else:
19                         vysledek = vysledek - 7
20                         pocet = pocet + 1
21
22
23                     if z3==0:
24                         vysledek = vysledek + 6
25                     else:
26                         vysledek = vysledek - 6
27                         pocet = pocet + 1
28
29
30                     if z4==0:
31                         vysledek = vysledek + 5
32                     else:
33                         vysledek = vysledek - 5
34                         pocet = pocet + 1
35
```

```
36
37                          if z5==0:
38                              vysledek = vysledek + 4
39                          else:
40                              vysledek = vysledek - 4
41                              pocet = pocet + 1
42
43
44                          if pocet == 2 and vysledek % 3 == 0:
45                              print("z1 = ", z1)
46                              print("z2 = ", z2)
47                              print("z3 = ", z3)
48                              print("z4 = ", z4)
49                              print("z5 = ", z5)
50                              print("vysledek = ", vysledek)
51                              print()
```

Program vypíše $(z_1, z_2, z_3, z_4, z_5, \mathit{vysledek})$: $(0, 0, 0, 1, 1, 21)$, $(0, 1, 0, 1, 0, 15)$, $(1, 0, 0, 0, 1, 15)$ a $(1, 1, 0, 0, 0, 9)$. Jednoduchými výpočty lze ověřit, že vypsaná řešení odpovídají všem podmínkám v zadání úlohy. Stejně tak lze ověřit správnost těchto výsledků v matematickém řešení úlohy na webových stránkách MO.

Následující úloha je zaměřena na práci s prvočísly.

Úloha 7 (viz [12])

Najděte všechny dvojice celých čísel x a y takových, že $x+y$ je prvočíslo a $3x + 5y$ je 16.

Autorské řešení. Ještě dříve než přistoupíme ke tvorbě funkčního programu úlohy, je žádoucí si připomenout základní rozdíl prvočísel a složených čísel. Prvočíslo je takové přirozené číslo, které je bez zbytku dělitelné pouze samo sebou a číslem 1. Například číslo 7 je prvočíslem, protože je beze zbytku dělitelné pouze čísly 1 a 7. Mezi přirozenými čísly 2 až 6 není žádné číslo, které by beze zbytku dělilo číslo 7. Naopak, číslo 6 není prvočíslem, protože mimo čísla 1 a 6 – mezi přirozenými čísly 2 až 5 – je ještě beze zbytku dělitelné čísly 2 a 3. Pokud tedy číslo N ($N > 2$) má být prvočíslem, pak žádné z přirozených čísel od 2 do $N - 1$ nesmí beze zbytku dělit číslo N. Této vlastnosti využijeme při konstrukci algoritmu pro testování prvočíselnosti.

V úloze se hledají všechna taková celá čísla x a y, pro která platí, že $3x + 5y = 16$ a zároveň číslo $x + y$ je prvočíslem, tj. $x + y > 1$.

Pro účely vyčíslení proměnných x a y uplatníme dva vzájemně vnořené cykly s pevným počtem opakování s označením řídících proměnných `x` a `y` (ř. 11 a 12). Ze zadání úlohy neplynou žádná omezení zdola i shora na čísla

x a y, je proto možné jejich číselné intervaly experimentálně nastavovat. V našem příkladu byl použit celočíselný interval od -100 do 100.

Z důvodu snížení výpočetní i časové náročnosti je vhodné nejprve v podmíněném příkazu posoudit podmínku $3x+5y = 16$ a zároveň (pomocí operátoru konjunkce) před započetím posuzování prvočíselnosti posoudit, zda $x+y$ je přirozeným číslem větší než 1 (ř. 14). Poté lze v dalším, vnořeném podmíněném příkazu posoudit prvočíselnost čísla $x+y$ (ř. 15). V případě kladného posouzení této podmínky lze vypsat číselné hodnoty `x` a `y`, neboť tvoří hledané řešení úlohy (ř. 16 a 17).

Nyní se ale vraťme k posouzení prvočíselnosti čísla $x+y$ (ř. 15), které je vhodné zformulovat v programu jako funkci. Tu vhodně pojmenujeme, např. `prv` s argumentem `z` (ř. 1). Budeme předpokládat, že číslo v `z` je prvočíslem. Zavedeme tedy pomocnou proměnnou, např. `prv`, kterou nastavíme na hodnotu `True`. Pokud v intervalu od 2 do `z` $-$ 1 (vyjádřeno pomocí cyklu s pevným počtem opakování – for cyklu (ř. 4)) bude nalezen alespoň jeden dělitel čísla v `z` (vyjádřeno podmíněným příkazem a s využitím operátoru modulo (ř. 5)), pak číslo v `z` nemůže být prvočíslem a proto proměnná `prv` bude přepsána na hodnotu `False`. Funkce vrátí hodnotu `True` v případě, že nebyl nalezen žádný dělitel, v opačném případě `False` (ř. 8).

```
1  def prv(z):
2      prv = True
3
4      for i in range(2, z):
5          if z % i == 0:
6              prv = False
7
8      return prv
9
10
11 for x in range(-100, 101):
12     for y in range(-100, 101):
13         soucet = x + y
14         if 3 * x + 5 * y == 16 and x + y > 1:
15             if prv(x + y)==True:
16                 print("x = ", x)
17                 print("y = ", y)
18                 print()
```

Program vypíše jako řešení jednu dvojici (x, y): $(-3, 5)$. Jednoduchým výpočtem lze ověřit, že vypsané řešení odpovídá všem podmínkám v zadání úlohy. Stejně tak lze ověřit správnost tohoto výsledku v matematickém řešení úlohy na webových stránkách MO.

Shrnutí

Prezentované úlohy mají sloužit k nastínění práce s matematickými úlohami MO, které lze řešit pomocí systematického experimentování. Jejich řešení mohou nalézt žáci střední i základní školy, kteří mají alespoň základní zkušenosti s programováním. Úmyslně proto nezařazujeme úlohy do konkrétních ročníků škol.

Článek sledoval dva cíle:

1. představit programování jako prostředek pro řešení matematických úloh MO, které lze řešit pomocí systematického experimentování;
2. ukázat na úlohách konkrétní algoritmické konstrukce řešení úloh.

Je vhodné zdůraznit, že představené úlohy nebyly řešeny z hlediska optimalizace (časové i paměťové), stejně tak nebyly použity pokročilejší algoritmické konstrukce.

Literatura

[1] Edu.cz: Revize RVP ICT. (2024). https://digitalizace.rvp.cz/g.

[2] *Eisenmann, P., Přibyl, J.*: Systematické experimentování ve výuce matematiky. In: Hašek, R. (ed.): Sborník příspěvků 6. konference Užití počítačů ve výuce matematiky, 7.–9. listopad 2013, Č. Budějovice, 2013, s. 85–93, https://home.pf.jcu.cz/~upvvm/2013/sbornik/clanky/10_UPVM2013_Eisenmann_Pribyl.pdf.

[3] *Hvorecký, J.*: Riešenie matematických úloh „hrubou silou“. In: Slavíčková, M. (ed.): Dva dni s didaktikou matematiky 2022. Zborník príspevkov, Univerzita Komenského, Bratislava, 2022, https://www.comae.sk/zbornik2022.pdf.

[4] *Kopka, J.*: Umění řešit matematické problémy. HAV, Praha, 2013.

[5] *Polya, G.*: Jak to řešit? Překvapivé aspekty (nejen) matematických metod. MatfyzPress, Praha, 2016.

[6] MO ČR (2023a), 73. ročník, kategorie Z9, krajské kolo, úloha 4.

[7] MO ČR (2017), 67. ročník, kategorie Z7, domácí kolo, úloha 1.

[8] MO ČR (2024a), 74. ročník, kategorie Z6, domácí kolo, úloha 3.

[9] MO ČR (2023b), 73. ročník, kategorie Z6, domácí kolo, úloha 3.

[10] MO ČR (2024c), 74. ročník, kategorie Z5, domácí kolo, úloha 4.

[11] MO ČR (2024e), 74. ročník, kategorie Z5, domácí kolo, úloha 3.

[12] MO ČR (2024f), 74. ročník, kategorie Z9, domácí kolo, úloha 1.