Ladislav Perk

Pojďme hledat řešení některých úloh matematické olympiády pomocí programování I

Rozhledy matematicko-fyzikální, Vol. 99 (2024), No. 1, 30–46

Persistent URL: `http://dml.cz/dmlcz/152336`

INFORMATIKA

Pojďme hledat řešení některých úloh matematické olympiády pomocí programování I

Ladislav Perk, PřF UJEP v Ústí nad Labem

Abstrakt. V předloženém příspěvku se zabýváme problematikou uplatnění programování při řešení vybraných úloh matematické olympiády ČR a SR. Na celkem šesti úlohách, ve kterých se pracuje s přirozenými čísly, jsou ukázány postupy řešení takových úloh s tím, že cílem námi prezentovaného způsobu řešení úloh je napsání funkčního programu v jazyce Python a získání shodných výsledků, které by bylo možné získat v rámci řešení matematickými postupy vyžadovanými v matematických úlohách. Součástí každé z těchto prezentovaných úloh jsou doplňkové úlohy, které si můžete vyřešit samostatně.

Úvod

Jedním ze zájmů soudobé matematiky je snaha přilákat a získat co nejvíce zájemců z řad žáků základních a středních škol a vzbudit tak u nich zájem o matematické vzdělávání. Využívá k tomu řadu nástrojů, jedním z nich je pořádání matematické olympiády v ČR a SR (dále jen MO), kterou lze pokládat za velmi úspěšnou popularizaci matematiky.

Možná se někteří z vás – kteří řešili úlohy MO [1] – dostali při řešení některé z úloh do situace, kdy jste tápali v dalším postupu a přišlo by vám vhod, kdybyste znali správné řešení úlohy. A právě pomocí programování toho u vybraných úloh můžete dosáhnout. U úloh, kde se pracuje s přirozenými, popř. celými čísly.

V tomto článku byly vybrány takové úlohy MO, kde se pracuje s přirozenými čísly a jsou hledána či posuzována pouze přirozená čísla. V těchto úlohách se nezaměřujeme na problematiku dělitelnosti přirozených čísel, tou se budeme zabývat v příštích článcích.

Řešení matematických úloh pomocí programování nemá v žádném případě plnohodnotně nahrazovat matematické řešení požadované v MO, jedná se pouze o jakýsi doplněk požadovaných matematických řešení.

Tento způsob můžete využít například v situaci, kdy nejste schopni nalézt řešení matematickými prostředky – vypsané výsledky vámi napsaného programu vám pak mohou napovědět při řešení matematickými prostředky. Dále pak můžete tento způsob využít v okamžiku, kdy si potřebujete zkontrolovat správnost vypočítaných výsledků.

Dalším z cílů je, abyste si uvědomili, že programováním jste získali další nástroj, jak řešit vybrané typy matematický úloh, a to nejen úlohy z MO.

Ukázky řešení vybraných úloh matematické olympiády

V tomto článku vám bude představeno celkem šest úloh MO, které lze vyřešit pomocí programování. Všechny tyto úlohy pracují s přirozenými čísly.

Každá úloha obsahuje text zadání úlohy přesně tak, jak je uveden v zadání MO. Po krátkém popisku následuje slovní výčet aktivit, které je nutné zrealizovat pro úspěšné napsání zdrojového kódu v jazyce Python. Každá z těchto aktivit je následně popsána podrobněji; je například zdůvodněno, jaký typ cyklu je vhodné pro vyčíslovávání proměnných použít, zda je či není vhodné pojmenování dalších pomocných proměnných, jakým způsobem se sestavuje podmínka či podmínky pro výpis čísel, které mají být splněny, jakým způsobem se tato čísla či jejich počet vypisuje apod.

Následuje zdrojový kód v jazyce Python, který vypisuje výsledky, které jsou shodné s výsledky uvedenými v MO. Můžete si napsat vlastní zdrojový kód či opsat zde předkládané zdrojové kódy. Je vhodné, abyste se zdrojové kódy pokusili napsat nejprve sami, pokud by se vyskytly komplikace, pak je možné opsat zde předkládaný zdrojový kód.

Na konci některých úloh jsou prezentovány doplňkové úlohy, u kterých nejsou uvedena jejich řešení. Řešení těchto úloh lze získat dílčí modifikací příslušného zdrojového kódu. Řešení těchto úloh budou publikována v dalším článku „Pojďme hledat řešení některých úloh matematické olympiády pomocí programování II“.

Dále je třeba zdůraznit, že se předpokládá, že máte alespoň základní zkušenosti s programováním v jazyce Python. Cílem tohoto článku není výuka jazyka Python, ale ukázka uplatnění programování v jazyce Python při řešení matematických úloh.

První úlohu lze pokládat za algoritmicky jednodušší. V rámci řešení je možné – pro účely prohledávání stavového prostoru – použít pouze jeden cyklus s pevným počtem opakování. Je však při řešení kladen důraz na vaši schopnost správných dekadických zápisů.

Úloha 1. *Zuzka napsala pětimístné číslo. Když připsala jedničku na konec tohoto čísla, dostala číslo, které je třikrát větší než číslo, které by* *získala, kdyby napsala jedničku před původní číslo. Které pětimístné číslo Zuzka napsala?* [2]

Řešení. Hledané pěticiferné číslo v programu označme proměnnou `cislo`.

Při tvorbě funkčního zdrojového kódu je zapotřebí se soustředit na následující kroky:

a) způsob vyčíslování proměnné `cislo`;

b) způsob zformulování šesticiferného čísla, které vznikne přidáním jedničky před pěticiferné číslo, stejně tak šesticiferného čísla, které vznikne přidáním jedničky za pěticiferné číslo;

c) posouzení týkající se vzájemné velikosti obou šesticiferných čísel;

d) výpis všech pěticiferných čísel v proměnné `cislo`, která splňují podmínku pro jejich výpis.

Nyní výše uvedené kroky rozebereme:

ad a) Protože se jedná o pěticiferné číslo, všechna přirozená čísla, která budeme posuzovat, budou v množině všech pěticiferných čísel od 10 000 do 99 999. Je tedy vhodné uplatnit cyklus s pevným počtem opakování (ř. 1).

ad b) Z původního pěticiferného čísla mají vzniknout dvě další šesticiferná čísla. Pro zpřehlednění kódu je vhodné každé z těchto čísel formulovat pomocí pomocné proměnné.

Šesticiferné číslo, které vznikne z původního pěticiferného čísla přidáním jedničky na konec, můžeme označit např. proměnnou `prvni` a vznikne tak, že se vynásobí deseti a přičte jednička (ř. 2).

Šesticiferné číslo, které vznikne z původního pěticiferného čísla přidáním jedničky na začátek, vznikne tak, že jedničku posuneme na místo statisíců a přičteme pěticiferné číslo.

To můžeme označit proměnnou `druhe` (ř. 3).

ad c) Splnění či nesplnění podmínky pro vypsání pěticiferného čísla jako hledaného řešení spočívá v pravdivostním ohodnocení výrazu. Musí platit, že číslo v `prvni` je třikrát větší než číslo v `druhe`. Proto výraz sestavíme tak, že trojnásobek čísla v `druhe` porovnáme s číslem v `prvni` (ř. 5).

ad d) V případě kladného vyhodnocení podmínky je možné vypsat číselnou hodnotu proměnné `cislo` (ř. 6).

```
for cislo in range(10000, 100000):
    prvni = 10 * cislo + 1
    druhe = 1 * 100000 + cislo

    if prvni == 3 * druhe:
        print("Nalezene cislo: ", cislo)
```

Program vypíše jedno pěticiferné číslo, a to 42 857. V řešení MO [2] je jako výsledek představeno číslo 42 857.

Druhou úlohu lze také pokládat za algoritmicky jednodušší. V rámci řešení je však nutné pro účely prohledávání stavového prostoru uplatnit tři vzájemně vnořené cykly s pevným počtem opakování.

Úloha 2. *Eva má tři papírky a na každém z nich je napsáno jedno přirozené číslo. Když vynásobí mezi sebou dvojice čísel z papírků, dostane výsledky 48, 192 a 36. Která čísla jsou napsána na Eviných papírcích?* [3]

Řešení. Hledaná tři přirozená čísla označme v programu proměnnými `a`, `b` a `c`.

Při tvorbě funkčního zdrojového kódu je zapotřebí se soustředit na následující kroky:

a) pojmenování tří proměnných, které reprezentují tři hledaná přirozená čísla napsaná na třech papírcích, a způsob jejich vyčíslení;
b) zformulování součinů dvojic čísel na papírcích;
c) posouzení, zda součiny přirozených čísel odpovídají číslům 48, 192 a 36;
d) výpis nalezených přirozených čísel, která mají být napsána na papírcích.

ad a) Protože je znám konečný počet přirozených čísel, kterých mohou proměnné `a`, `b` a `c` nabývat, je vhodné pro jejich vyčíslovávání použít tři vzájemně vnořené cykly s pevným počtem opakování.

Bez dalších úvah o číselných omezeních proměnných `a`, `b` a `c` lze uvažovat, že každá z těchto proměnných může být přirozeným číslem od 1 do 192 (ř. 1–3).

ad b) Příslušné součiny proměnných `a`, `b` a `c` lze vhodně označit, například pomocí proměnných `ab`, `bc` a `ac` (ř. 4–6).

ad c) Posouzení, zda příslušné součiny čísel v proměnných `a`, `b` a `c` odpovídají číslům 48, 192 a 36, lze provést pomocí podmínky `if`, a to v konjunktivním tvaru za použití `and`. (ř. 8). Stejně tak lze dílčí podmínky zapsat pomocí tří podmíněných příkazů `if` zapsaných pod sebe.

ad d) Po splnění výše uvedené podmínky je vhodné provést výpis nalezených trojic přirozených čísel (ř. 9).

```
1  for a in range(1, 193):
2      for b in range(1, 193):
3          for c in range(1, 193):
4              ab = a * b
5              bc = b * c
6              ac = a * c
7
8              if ab == 48 and bc == 192 and ac == 36:
9                  print("Nalezena trojice: ", a, ",", b, ",", c)
10                 print()
```

Program jako řešení vypíše jedno trojčíslí: (3, 16, 12). V MO [3] jsou jako řešení uvedena tři přirozená čísla 16, 3 a 12.

Protože nezáleží na pořadí zápisu čísel napsaných papírcích, lze obě řešení považovat za shodná.

Nalezněte řešení níže uvedených tří doplňkových úloh dílčími úpravami vámi vytvořeného programu:

Doplňkové úlohy

2.1. *Eva má čtyři papírky a na každém z nich je napsáno jedno přirozené číslo. Když vynásobí mezi sebou všechny trojice čísel z papírků, dostane výsledky 30, 36, 60 a 90. Která čísla jsou napsána na Eviných papírcích?*

2.2. *Eva má čtyři papírky a na každém z nich je napsáno jedno přirozené číslo. Když vynásobí mezi sebou všechny dvojice čísel z papírků, dostane výsledky 6, 14, 10, 21, 15 a 35. Která čísla jsou napsána na Eviných papírcích?*

2.3. *Eva má pět papírků a na každém z nich je napsáno jedno přirozené číslo. Když vynásobí mezi sebou všechny čtveřice čísel z papírků, dostane výsledky 24, 48, 16, 12 a 24. Která čísla jsou napsána na Eviných papírcích?*

Třetí úlohu lze – stejně jako předchozí dvě – pokládat za algoritmicky jednodušší. V rámci řešení je však nutné pro účely prohledávání stavového prostoru uplatnit tři vzájemně vnořené cykly s pevným počtem opakování a správně zformulovat tři dílčí součiny tří přirozených čísel.

Úloha 3. Zadání úlohy v MO:
Součin tří přirozených čísel je 600. Kdybychom jednoho činitele zmenšili o 10, zmenšil by se součin o 400. Kdybychom místo toho jednoho činitele zvětšili o 5, zvětšil by se součin na dvojnásobek původní hodnoty. Která tři přirozená čísla mají tuto vlastnost? [4]

Řešení. Hledaná tři přirozená čísla označme v programu proměnnými `a`, `b` a `c`.

Při tvorbě funkčního zdrojového kódu je zapotřebí se soustředit na následující kroky:

a) způsob vyčíslování proměnných `a`, `b` a `c`;

b) zformulování dílčích podmínek, které musí dané trojice přirozených čísel splňovat;

c) zajištění výpisů nalezených trojic přirozených čísel, které splňují všechny podmínky pro jejich výpis.

Nyní výše uvedené kroky rozebereme:

ad a) Protože ze zadání plyne, že součin čísel v `a`, `b` a `c` je roven číslu 600, pak každé z čísel v `a`, `b` a `c` může mít číselnou hodnotu nejvýše 600 (za předpokladu, že právě dvě libovolné proměnné budou jedničky). A protože se jedná o přirozená čísla, pak čísla v `a`, `b` a `c` mají velikost od 1 do 600. Z tohoto důvodu je vhodné uplatnit tři vzájemné cykly s pevným počtem opakování (ř. 1–3).

ad b) Podmíněné vykonání příkazu v podobě výpisů číselných hodnot proměnných `a`, `b` a `c` je tvořeno podmínkou, která obsahuje logický výraz. Tento výraz je tvořen třemi dílčími výrazy: výraz je vyhodnocen jako pravdivý, pokud všechny tři dílčí výrazy jsou vyhodnoceny jako pravdivé.

Dílčími výrazy jsou:

- prvním je `a * b * c == soucin`;
- druhým je `(a - 10) * b * c == soucin - 400`;
- a třetím je `a * (b + 5) * c == 2 * soucin`.

Protože všechny tři dílčí výrazy musí být splněny, nabízí se přinejmenším dvě možnosti zápisu: buď použití vzájemně vnořených příkazů `if` pro každý dílčí výraz zvlášť, nebo použití jednoho podmíněného příkazu `if`, jehož podmínku budou tvořit tři dílčí podmínky v konjuktivním tvaru, tzn. mezi dílčími výrazy se použije operátor `and`. Je vhodné, abyste použili spíše druhou z možností (ř. 6).

Uveďme zdůvodnění: pokud bychom každou z dílčích podmínek formulovali zvlášť pomocí vzájemně vnořených příkazů `if` „pod sebe“, sice bychom zdrojový kód mírně zpřehlednili, ale v případě, že pro každý `if` nebude formulováno alternativní řešení v případném `else` (popř. `elif`), pak takové „větvení“ zdrojového kódu nemá smysl. Navíc je tento způsob neúsporný k pracovnímu prostoru.

V zadání úlohy ale není jednoznačně stanoveno, zda se zvětšení o 10 a zmenšení o 5 vztahuje na dva různé činitele či právě jednoho činitele. Proto je zapotřebí uvažovat také variantu, kdy právě jednoho z činitelů zmenšíme o 10, stejně tak zvětšíme o 5. Lze si libovolně vybrat některou z proměnných `a`, `b` a `c`; použijme například proměnnou `a`. Výsledné posouzení (ř. 10) vznikne analogicky jako předchozí posouzení (ř. 6).

ad c) Výpisy hodnot proměnných `a`, `b` a `c` je nutné realizovat pro každé posouzení zvlášť (ř. 7 a 11).

```
1  for a in range(1, 601):
2      for b in range(1, 601):
3          for c in range(1, 601):
4              soucin = 600
5
6              if a * b * c == soucin and (a - 10) * b * c ==
                   soucin - 400 and a * (b + 5) * c == 2 *
                   soucin:
7                  print("Nalezena trojice: ", a, ",", b, ",", c)
8                  print()
9
10             if a * b * c == soucin and (a - 10) * b * c ==
                   soucin - 400 and (a + 5) * b * c == 2 *
                   soucin:
11                 print("Nalezena trojice: ", a, ",", b, ",", c)
12                 print()
```

Program vypíše tři přirozená čísla 8, 5, 15. V řešení MO [4] je jako řešení uvedena trojice čísel 8, 15 a 5.

Nalezněte řešení níže uvedených tří doplňkových úloh dílčími úpravami vámi vytvořeného programu:

Doplňkové úlohy

3.1. *Součin čtyř přirozených čísel je 120. Kdybychom první z nich zvětšili o 5, zvětšil by se součin o 200. Kdybychom druhé z nich zvětšili o 5, zvětšil by se o 150. Kdybychom třetí z nich zmenšili o 1, zmenšil by se o 60. Kdybychom čtvrté z nich zvětšili o 10, zvětšil by se na trojnásobek původní hodnoty. Která čtyři přirozená čísla mají tuto vlastnost?*

3.2. *Součin pěti přirozených čísel je 72. Kdybychom první z nich zmenšili o 2, zmenšil by se součin o 48. Kdybychom druhé z nich zvětšili o 4, zvětšil by se o 288. Kdybychom třetí z nich zmenšili o 3, zmenšil by se o 54. Kdybychom čtvrté z nich zvětšili o 6, zvětšil by se na dvojnásobek původní hodnoty. Která pětice přirozených čísel má tuto vlastnost?*

3.3. *Součin tří vzestupně seřazených, vzájemně různých, přirozených čísel je 1 001. Kdybychom nejmenší z nich zvětšili o 20, zvětšil by se součin o 2 860. Kdybychom největší z nich zmenšili o 5, zmenšil by se součin o 385. Kdybychom prostřední ze tří čísel zvětšili o 33, zvětšil by se součin na čtyřnásobek. Která přirozená čísla mají tuto vlastnost?*

Čtvrtou úlohu lze již pokládat za algoritmicky náročnější. Pracuje se zde s ciframi, resp. dvojciframi hledaných čtyřciferných čísel.

Úloha 4. *Uvažme čtyřmístné přirozené číslo s následující vlastností: jestliže prohodíme jeho první dvojčíslí s druhým, dostaneme čtyřmístné číslo o 99 menší. Kolik je takových čísel celkem?* [5]

Hledané čtyřciferné číslo označme v programu proměnnou `abcd`, jeho první dvojčíslí `ab` a jeho druhé dvojčíslí `cd`.

Při tvorbě funkčního zdrojového kódu je zapotřebí se soustředit na následující kroky:

a) způsob vyčíslování proměnných `abcd`, `ab` a `cd`;
b) zformulování hledaného čtyřciferného čísla a čtyřciferného čísla s prohozeným dvojčíslím pomocí dekadických zápisů;
c) posouzení, zda čtyřčíslí s prohozeným dvojčíslím je o 99 menší než hledané čtyřčíslí;
d) zajištění výpisů nalezených čtyřciferných čísel v `abcd`, která splňují podmínku pro jejich výpis.

Nyní výše uvedené kroky rozebereme:

ad a) Každé dvojčíslí označené proměnnými `ab` a `cd` postupně nabývá číselných hodnot od 10 do 99, proto pro jejich vyčíslování je vhodné použít dva vzájemně vnořené cykly s pevným počtem opakování (ř. 3 a 4).

ad b) Zformulování čísel v proměnných `abcd` a `dcba` pomocí proměnných `ab` a `cd` je možné pomocí dekadických zápisů. Číslo `abcd` vytvoříme tak, že dvojčíslí v `ab` vynásobíme stovkou a přičteme dvojčíslí v `cd` (ř. 5). Analogicky číslo v `cdab` vytvoříme tak, že dvojčíslí v `cd` vynásobíme stovkou a přičteme dvojčíslí v `ab` (ř. 6).

ad c) Protože číslo v `cdab` je o 99 menší než číslo v `abcd`, pak k číslu v `cdab` přičteme 99, aby se čísla v `abcd` a v `cdab` rovnala a mohli jsme je v podmíněném příkazu porovnat (ř. 8).

ad d) Výhodou tohoto kódu je, že si můžeme konkrétní čtyřciferná čísla `abcd` splňující podmínky vypsat (ř. 9). Stanovení počtu těchto čísel je realizováno inkrementací proměnné `pocet` (ř. 10), kterou je nutné před začátkem hledání vynulovat (ř. 1). Vlastní výpis počtu čísel je nutné vypsat mimo konstrukci hledání (ř. 12).

```
1  pocet = 0
2
3  for ab in range(10, 100):
4      for cd in range(10, 100):
5          abcd = ab * 100 + cd
6          cdab = cd * 100 + ab
7
8          if abcd == cdab + 99:
9              print(abcd)
10             pocet = pocet + 1
11
12 print("Nalezeny pocet: ", pocet)
```

Program vypíše počet hledaných čtyřciferných čísel, a to číslo 89. Dále pak těchto 89 čísel: 1 110, 1 211, 1 312, 1 413, 1 514, 1 615, 1 716, 1 817, 1 918, 2 019, 2 120, 2 221, 2 322, 2 423, 2 524, 2 625, 2 726, 2 827, 2 928, 3 029, 3 130, 3 231, 3 332, 3 433, 3 534, 3 635, 3 736, 3 837, 3 938, 4 039, 4 140, 4 241, 4 342, 4 443, 4 544, 4 645, 4 746, 4 847, 4 948, 5 049, 5 150, 5 251, 5 352, 5 453, 5 554, 5 655, 5 756, 5 857, 5 958, 6 059, 6 160, 6 261, 6 362, 6 463, 6 564, 6 665, 6 766, 6 867, 6 968, 7 069, 7 170, 7 271, 7 372, 7 473, 7 574, 7 675, 7 776, 7 877, 7 978, 8 079, 8 180, 8 281, 8 382, 8 483, 8 584, 8 685, 8 786, 8 887, 8 988, 9 089, 9 190, 9 291, 9 392, 9 493, 9 594, 9 695, 9 796, 9 897, 9 998. V řešení MO [5] je uveden počet hledaných čtyřciferných čísel 89.

Nalezněte řešení níže uvedených tří doplňkových úloh dílčími úpravami vámi vytvořeného programu:

Doplňkové úlohy

4.1. *Uvažme pětimístné přirozené číslo s následující vlastností: jestliže prohodíme jeho první trojčíslí s posledním dvojčíslím, dostaneme pětimístné číslo o 14 769 menší. Kolik je takových čísel celkem?*

4.3. *Uvažme šestimístné přirozené číslo s následující vlastností: jestliže prohodíme jeho první trojčíslí s druhým trojčíslím, dostaneme šestimístné číslo o 885 114 menší. Kolik je takových čísel celkem?*

4.3. *Uvažme sedmimístné přirozené číslo s následující vlastností: jestliže prohodíme jeho první trojčíslí s posledním čtyřčíslím, dostaneme sedmimístné číslo o 8 515 341 větší. Kolik je takových čísel celkem?*

V páté úloze se budeme zabývat hledáním takového čtyřciferného čísla, jehož cifry nejenže musí splňovat určité vlastnosti, ale zároveň také bude číslem, které bude charakterizované největším rozdílem své hodnoty vůči hodnotě svého opačně ciferně psaného čísla. K tomu uplatníme algoritmickou konstrukci, která je známa jako hledání maximálního prvku. Řešení úlohy lze považovat za náročnější.

Úloha 5. *Monika přemýšlí o čtyřmístném čísle, které má následující vlastnosti:*

- *součin dvou krajních číslic je 40,*
- *součin dvou vnitřních číslic je 18,*
- *rozdíl dvou krajních číslic je stejný jako rozdíl dvou vnitřních číslic,*
- *rozdíl myšleného čísla a opačně napsaného čísla (tj. čísla napsaného stejnými číslicemi, ale v opačném pořadí) je největší možný.*

Určete Moničino myšlené číslo. [6]

Řešení. Jednotlivé cifry hledaného čtyřciferného přirozeného čísla označme v programu proměnnými `a` (tisíce), `b` (stovky), `c` (desítky) a `d` (jednotky).

Při tvorbě funkčního zdrojového kódu je zapotřebí se soustředit na následující kroky:

a) vyčíslení proměnných `a` až `d`;

b) zformulování hledaného čtyřciferného čísla a jeho opačně napsaného čísla pomocí cifer v proměnných `a` až `d`;

c) posouzení splnění všech podmínek kladených na jednotlivé cifry `a` až `d`;

d) zformulování způsobu nalezení čtyřciferného čísla s největším rozdílem vůči svému opačně zapsanému číslu;
e) výpis hledaného čtyřciferného čísla, které splňuje výše uvedené podmínky.

Nyní výše uvedené kroky rozebereme:

ad a) Proměnné `a`, `b`, `c` a `d` reprezentují jednotlivé cifry hledaného čtyřciferného čísla. Ze zadání plyne, že součiny dvojic cifer jsou nenulové, pak všechny proměnné `a` až `d` nebudou nabývat hodnoty 0. Proto se uplatní vzájemně vnořené cykly s pevným počtem opakování s vyčíslením cifer v `a` až `d` od 1 do 9 (ř. 3–6).

ad b) Hledané čtyřciferné číslo označíme s využitím označení proměnných `abcd`, jeho ciferně opačné číslo pak `dcba`; pro obě čísla pak vyjádření vyčíslení pomocí příslušných dekadických zápisů (ř. 7 a 8).

ad c) Posouzení velikosti součinu dvou krajních číslic lze zapsat výrazem `a * d == 40`, posouzení velikosti součinu dvou vnitřních číslic pak `b * c == 18`.

Rozdíl dvou krajních číslic neřeší, zda má být vyčísleno `a` − `d` či `d` − `a`, stejně tak u rozdílu dvou vnitřních číslic `b` − `c` či `c` − `b`. Číselné hodnoty jsou stejné, liší se jen znaménkem, proto je vhodné uplatnit absolutní hodnotu ve vyčíslovávání rozdílů; výsledný výraz lze potom zapsat `abs(a - d) == abs(b - c)` (ř. 10).

ad d) Nyní se máme začít zabývat nalezením takového čtyřciferného čísla, které má se svým opačně napsaným číslem největší rozdíl. V programování se jedná o nalezení největšího čísla (maxima) z určité konečné neprázdné množiny čísel. Pro ty z vás, kteří s touto algoritmickou konstrukcí nemají zkušenosti, jsou určeny následující řádky tohoto bodu.

Pro účely hledání největšího z čísel `abcd` − `dcba` je třeba toto číslo nejprve vyjádřit pomocí vhodné proměnné, např. `rozdil` (ř. 11). Zároveň se zavede pomocná proměnná, např. `nejvetsi`, která registruje největší číslo ze všech posouzených čísel (v našem případě rozdílů v proměnné `rozdil`), po ukončení hledání pak hledaný největší rozdíl. Zároveň se zavede proměnná `cislo`, která bude registrovat číslo v proměnné `abcd` (ř. 15).

Princip hledání největšího z čísel `rozdil` lze ilustrovat následujícím zápisem:

$$\texttt{nejvetsi} := \begin{cases} \texttt{rozdil}, & \text{jestliže } \texttt{nejvetsi < rozdil}; \\ \texttt{nejvetsi}, & \text{jestliže } \texttt{nejvetsi >= rozdil}. \end{cases}$$

V případě, že číslo v proměnné `rozdil` při určitých hodnotách `a`, `b`, `c` a `d` je větší než číslo v proměnné `nejvetsi` (ř. 13), pak se hodnota v proměnné `nejvetsi` přepíše na číselnou hodnotu v proměnné `rozdil` (ř. 14). Zároveň se zapíše do proměnné `cislo` číselná hodnota v `abcd` (ř. 15). V opačném případě k žádnému přepisu číselné hodnoty v proměnných `nejvetsi` a `cislo` nedochází.

Pro každou ze všech číselných hodnot v `a`, `b`, `c` a `d` se tedy vyčíslí proměnná `rozdil` a porovná s číselnou hodnotou v `nejvetsi`, která je přepisována výše uvedeným způsobem. Je tedy zřejmé, že po posouzení všech číselných hodnot v `a`, `b`, `c` a `d` je v proměnné `nejvetsi` zapsána číselná hodnota hledaného největšího číselného rozdílu a v proměnné `cislo` pak číslo v `abcd`, při které byla největší hodnota číselného rozdílu nalezena.

Je třeba zdůraznit, že proměnnou `rozdil` je nutné před započetím hledání nastavit na takovou hodnotu, kdy platí, že všechny číselné hodnoty rozdílů budou mít vždy větší hodnotu než tato nastavená číselná hodnota. Proto proměnnou `rozdil` nastavíme na hodnotu např. $-99\,999$ (ř. 1).

ad e) Výpis číselné hodnoty v proměnné `cislo`, která je hledaným čtyřciferným číslem, je nutné zrealizovat až po ukončení prohledávání stavového prostoru (ř. 17).

```
1  nejvetsi = -99999
2
3  for a in range(1, 10):
4      for b in range(1, 10):
5          for c in range(1, 10):
6              for d in range(1, 10):
7                  abcd = a * 1000 + b * 100 + c * 10 + d
8                  dcba = d * 1000 + c * 100 + b * 10 + a
9
10                 if a * d == 40 and b * c == 18 and abs(a
                       - d) == abs(b - c):
11                     rozdil = abcd - dcba
12
13                     if rozdil > nejvetsi:
14                         nejvetsi = rozdil
15                         cislo = abcd
16
17 print("Nalezene cislo: ", cislo)
```

Program vypíše čtyřciferné přirozené číslo 8 635. V řešení MO [6] je uvedeno čtyřciferné číslo 8 635.

Nalezněte řešení níže uvedených tří doplňkových úloh dílčími úpravami vámi vytvořeného programu:

Doplňkové úlohy

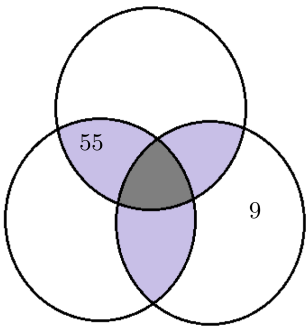
5.1. *Monika přemýšlí o největším pětimístném čísle, které má následující vlastnosti: součin prvních dvou a posledních dvou cifer je 448; cifra uprostřed je menší než 6; rozdíl cifer první dvojice cifer je dvakrát větší než rozdíl cifer poslední dvojice cifer; rozdíl myšleného čísla a opačně napsaného čísla je největší možný. Určete Moničino myšlené číslo.*

5.2. *Monika přemýšlí o největším šestimístném čísle, které má následující vlastnosti: součin prvních dvou a posledních dvou cifer je 2 160; součin cifer ve druhé dvojici cifer je 21; rozdíl cifer první dvojice cifer je stejný jako rozdíl cifer poslední dvojice cifer; rozdíl myšleného čísla a opačně napsaného čísla je největší možný. Určete Moničino myšlené číslo.*

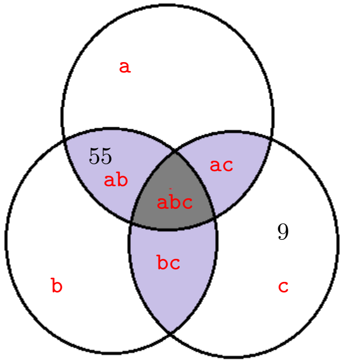
5.3. *Monika přemýšlí o čtyřmístném čísle, které má následující vlastnosti: součin dvou krajních číslic je 30; součin dvou vnitřních číslic je 56; rozdíl dvou krajních číslic je stejný jako rozdíl dvou vnitřních číslic. Ve čtyřciferném čísle vyškrtneme vždy jednu číslici a získáme tak čtyři trojciferná čísla. Pro Moničino číslo platí, že součet těchto trojciferných čísel je největší možný. Určete Moničino myšlené číslo.*

Šestou – a poslední – úlohu lze považovat za algoritmicky jednodušší. Je však specifická tím, že vlastnosti, které mají splňovat hledaná přirozená čísla, jsou formulovány grafickým způsobem.

Úloha 6. *Do prázdných polí v následujícím obrázku doplňte celá čísla větší než 1 tak, aby v každém tmavším políčku byl součin čísel ze sousedních světlejších políček: Jaké je číslo ve středu?* [7]



Řešení. Předpokládejme označení jednotlivých políček podle níže uvedených červeně označených proměnných, které použijeme v programu.



Při tvorbě funkčního zdrojového kódu je zapotřebí se soustředit na následující kroky:

a) způsob vyčíslování proměnných `a` a `b`;
b) posouzení, zda součin příslušných hledaných proměnných je roven číslu 55;
c) zformulování proměnných, které reprezentují číselné hodnoty ve všech zbylých neoznačených políčcích;
d) zajištění výpisů nalezených přirozených čísel v `abc`, která splňují všechny podmínky pro jejich výpis.

Ještě dříve, než přistoupíme k rozboru řešení úlohy, vysvětlíme si význam „sousednosti" políček. Sousednost políček znamená, že hodnota čísla v tmavším políčku vznikne jako součin čísel v těch světlejších políčcích, které právě toto tmavší políčko vzájemně sdílejí.

Podívejme se na obrázek výše. Číslo 55 v tmavším políčku vznikne jako součin čísel ve světlejších políčcích, které lze označit proměnnými `a` a `b`. Čísla v `a` a `b` spolu sousedí a vzájemně sdílejí tmavší políčko, jehož číselnou hodnotu lze registrovat v proměnné `ab` a ze zadání úlohy má konstantní hodnotu, číslo 55.

ad a) Protože součin čísel `a` a `b` je dle zadání roven číslu 55, pak proměnné `a` a `b` mohou postupně nabývat číselných hodnot od 2 do 55 (ř. 2 a 3).

ad b) Posouzení, zda součin čísel v `a` a `b` je roven číslu 55, lze zrealizovat pomocí podmíněného příkazu `if` (ř. 4).

ad c) Zbylá neoznačená políčka lze označit proměnnými, které odpovídají příslušným průnikům proměnných `a`, `b` a `c` (ř. 5–8).

ad d) Výpis číselných hodnot proměnných `a`, `b` a konstanty `c` (ř. 1–3) je nutné zrealizovat po podmíněném příkazu `if` (ř. 4) a formulování proměnných `ab`, `bc` a `ac` (ř. 5–7), stejně tak `abc` (ř. 8).

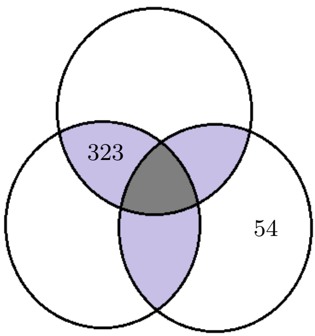
```
1  c = 9
2  for a in range(2, 56):
3      for b in range(2, 56):
4          if a * b == 55:
5              ab = a * b
6              bc = b * c
7              ac = a * c
8              abc = ab * bc * ac
9
10             print("Cislo v nejtmavsim policku: ", abc)
11             print("Cisla v nejsvetlejsich policcich: ", a, b, c)
12             print()
```

Program vypíše čísla (`a`, `b`, `c`): (5, 11, 9) a (11, 5, 9) a u obou těchto trojic číslo nacházející se v nejtmavším políčku, a to pro obě trojice číslo 245 025. V řešení MO [7] jsou nalezena zbylá čísla ve světlých polích 5 a 11, číslo v nejtmavším políčku je 245 025.

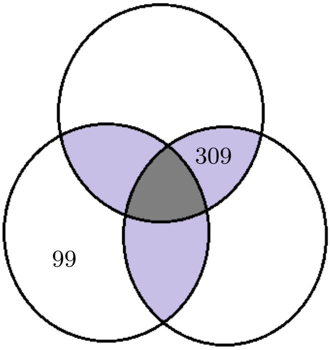
Nalezněte řešení níže uvedených tří doplňkových úloh dílčími úpravami vámi vytvořeného programu:

Doplňkové úlohy

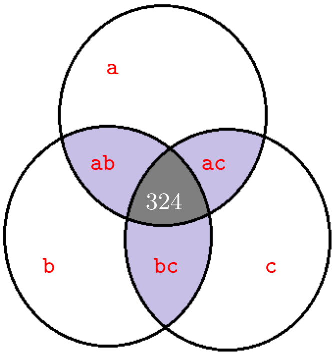
6.1. *Do prázdných polí v následujícím obrázku doplňte celá čísla větší než 1 tak, aby v každém tmavším políčku byl součin čísel ze sousedních světlejších políček: Jaké je číslo ve středu?*



6.2. *Do prázdných polí v následujícím obrázku doplňte celá čísla větší než 1 tak, aby v každém tmavším políčku byl součin čísel ze sousedních světlejších políček: Jaké je číslo ve středu?*



6.3. *Do prázdných polí v následujícím obrázku doplňte celá čísla větší než 1 tak, aby v každém tmavším políčku byl součin čísel ze sousedních světlejších políček a číslo ve středu bylo 324. Jaká čísla mohou být v proměnných* `a`, `b` *a* `c`*?*



Poděkování

Touto cestou bych rád poděkoval paní doc. Ing. Ľubomíře Dvořákové, Ph.D., a také pánům doc. RNDr. Pavlu Töpferovi, CSc., a PhDr. Jiřímu Přibylovi, Ph.D., za inspirativní připomínky k tomuto textu.

Literatura

[1] `https://www.matematickaolympiada.cz/`

[2] MO ČR, 58. ročník, kategorie Z7, domácí kolo, úloha 5. [online].[cit. 25.11.2023]. Dostupné na: `https://www.matematickaolympiada.cz/media/3491964/z58i-7.pdf`.

[3] MO ČR, 62. ročník, kategorie Z6, domácí kolo, úloha 4. [online]. [cit. 25.11.2023]. Dostupné na: `https://www.matematickaolympiada.cz/media/3492013/z62i-6.pdf`.

[4] MO ČR, 62. ročník, kategorie Z8, domácí kolo, úloha 1. [online]. [cit. 25.11.2023]. Dostupné na: `https://www.matematickaolympiada.cz/media/3492015/z62i-8.pdf`.

[5] MO ČR, 69. ročník, kategorie Z9, okresní kolo, úloha 3. [online]. [cit. 25.11.2023]. Dostupné na: `https://www.matematickaolympiada.cz/media/3492097/z69ii-9.pdf`.

[6] MO ČR, 66. ročník, kategorie Z8, okresní kolo, úloha 1. [online]. [cit. 25.11.2023]. Dostupné na: `https://www.matematickaolympiada.cz/media/3492058/z8ii-r.pdf`.

[7] MO ČR, 66. ročník, kategorie Z6, domácí kolo, úloha 6. [online]. [cit. 25.11.2023]. Dostupné na: `https://www.matematickaolympiada.cz/media/3492051/z66-6.pdf`.

Computer science is no more about computers than astronomy is about telescopes.

Informatika není o počítačích o nic víc, než je astronomie o dalekohledech.

I don’t need to waste my time with a computer just because I am a computer scientist.

Nemusím ztrácet svůj čas prací u počítače jen proto, že jsem informatik.

Edgar Dijkstra