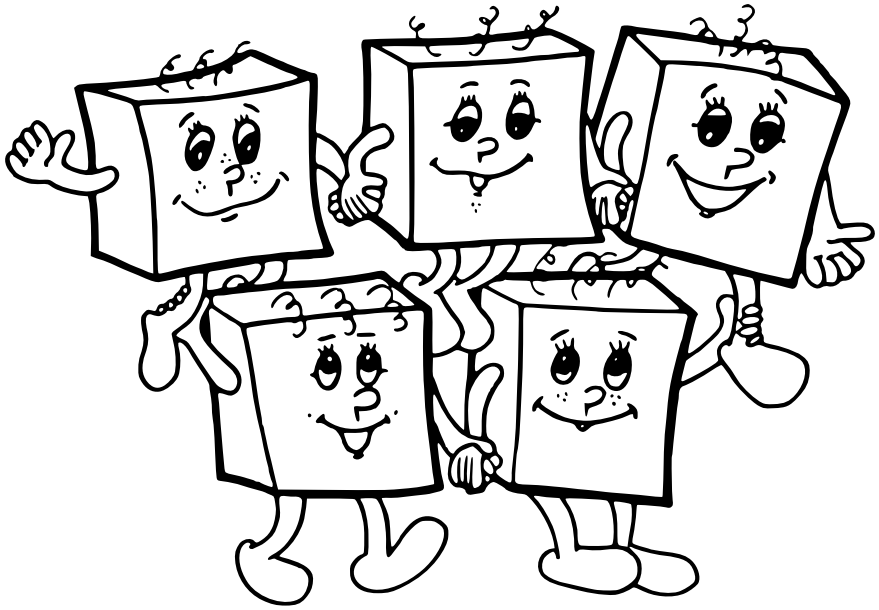
# OLYMPIÁDA V INFORMATIKE

## NA STREDNÝCH ŠKOLÁCH



**23. ročník**

školský rok 2007/08

**zadania domáceho kola**
**kategória A**

- **Olympiáda v informatike** je od školského roku 2006/07 samostatnou súťažou. Predchádzajúcich 21 ročníkov tejto súťaže prebiehalo pod názvom **Matematická olympiáda, kategória P** (programovanie).
- Oficiálnu **webstránku** súťaže nájdete na `http://www.ksp.sk/oi/`.

# Informácie a pravidlá

## Pre koho je súťaž určená?

OI sa uskutočňuje v týchto kategóriách:

a) **kategória A** je určená pre žiakov tretieho a štvrtého ročníka stredných škôl a príslušných ročníkov viacročných gymnázií a má tri kolá: domáce, krajské a celoštátne
b) **kategória B** je určená pre žiakov prvého a druhého ročníka stredných škôl a príslušných ročníkov viacročných gymnázií a má dve kolá: domáce a krajské

Do každej kategórie sa môžu zapojiť aj žiaci nižších ročníkov ako tí, pre ktorých je určená.

## Ako bude súťaž prebiehať?

V *domácom kole* každej kategórie účastníci riešia štyri úlohy. Riešenia odovzdajú svojmu učiteľovi informatiky do **15. novembra 2007**. Učitelia informatiky odošlú riešenia v tomto termíne zo školy na adresu príslušnej krajskej komisie OI. Adresy krajských komisií sú uvedené na poslednej strane tohto letáku.

Najúspešnejší riešitelia domáceho kola sú pozvaní do *krajského kola*, kde riešia štyri teoretické úlohy.

V kategórii A sú do *celoštátneho kola* pozývaní najúspešnejší riešitelia krajských kôl. Presnejšie, po vyhodnotení krajských kôl prebehne koordinácia bodovacích škál, spoja sa výsledkové listiny do jednej celoštátnej, a do celoštátneho kola sú pozvaní najlepší riešitelia podľa tejto výsledkovej listiny.

V celoštátnom kole účastníci prvý deň riešia tri teoretické úlohy, druhý deň dve praktické úlohy (pri počítači). Z najlepších riešiteľov tohto kola SK OI vyberie družstvá pre Medzinárodnú informatickú olympiádu (IOI) a Stredoeurópsku informatickú olympiádu (CEOI).

### Ako majú vyzerať riešenia domáceho kola?

Riešenia súťažných úloh domáceho kola pozostávajú z dvoch častí:

**Popis riešenia.** Riešenia musia obsahovať podrobný popis použitého algoritmu, **zdôvodnenie jeho správnosti** a diskusiu o efektivite zvoleného riešenia (t.j. posúdenie časových a pamäťových nárokov programu). Algoritmus by mal byť jasný už z popisu riešenia, teda bez toho, aby bolo potrebné nahliadnuť do programu.

**Program.** V úlohách 1, 2 a 3 je potrebné k riešeniu pripojiť odladený program napísaný v jazyku Pascal, C alebo C++. (Musí sa dať skompilovať kompilátorom *FreePascal*, resp. *gcc*.) Program sa odovzdáva v písomnej forme (vytlačený) aj v elektronickej forme (na CD, prípadne diskete).

Svoje programy pomenujte $k$`-i-`$x$`.pas/.c/.cpp`, kde $k$ je kategória a $x$ je číslo súťažnej úlohy. Z jednej školy možno poslať všetky riešenia na jednom CD. V takomto prípade pre každého riešiteľa vytvorte podadresár označený jeho priezviskom.

V úlohe 4 odovzdávate program len v písomnej podobe.

Popis riešenia vypracujte čitateľne na listy formátu A4. **Každú úlohu začnite na novom liste** a v záhlaví uveďte vaše meno, ročník, adresu školy a označenie príkladu podľa tohto letáku. Zadania úloh nemusíte opisovať. Ak sa vám riešenie úlohy nezmestí na jeden list, uveďte na ďalších listoch vľavo hore svoje meno a označenie úlohy, listy očíslujte a zopnite.

### Usporiadateľ súťaže

Olympiádu v informatike (OI) vyhlasuje *Ministerstvo školstva SR* v spolupráci so *Slovenskou informatickou spoločnosťou* a *Slovenskou komisiou Olympiády v informatike*. Súťaž organizuje *Slovenská komisia OI* a v jednotlivých krajoch ju riadia *krajské komisie OI*. Na jednotlivých školách ju zaisťujú učitelia informatiky. Celoštátne kolo OI, tlač materiálov a ich distribúciu po organizačnej stránke zabezpečuje IUVENTA v tesnej súčinnosti so Slovenskou komisiou OI.

# Zadania kategórie A

Do **15. novembra 2007** odovzdajte riešenia týchto úloh svojmu učiteľovi informatiky. Ten ich následne odošle na adresu príslušnej krajskej komisie OI. Adresy krajských komisií sú uvedené na poslednej strane tohto letáku.

## A-I-1 O zdanlivom kopci

Miško a Jožko sa chystali na výlet. Vymysleli si trasu, kadiaľ spolu pôjdu, potom každý vytiahol svoju mapu a začal si trasu pozerať. Niektoré body, cez ktoré trasa viedla, mali na mapách zapísanú nadmorskú výšku.

„To je zaujímavé," povedal po chvíli Miško. „Keď sa pozerám len na nadmorské výšky bodov, cez ktoré pôjdeme, vyzerá to, že ideme len cez jeden kopec: najskôr hore, potom dole."

„Ale kdeže, to nie je pravda." zadivil sa Jožko.

Netrvalo dlho, kým zistili, kde vznikol problém: mali rôzne presné mapy. Niektoré výšky, ktoré boli na Jožkovej mape vyznačené, na Miškovej chýbali.

### Súťažná úloha

Napíšte program, ktorý dostane na vstupe zoznam výšok na Jožkovej mape a zistí, koľko najviac z nich môže byť vyznačených aj na Miškovej mape.

### Formát vstupu

Prvý riadok vstupu obsahuje jedno celé číslo $N$ ($1 \leq N \leq 100\,000$) – počet takých bodov na trase výletu, ktoré majú na Jožkovej mape vyznačenú nadmorskú výšku. Ďalšie riadky obsahujú dokopy $N$ celých čísel z rozsahu 1 až $1\,000\,000\,000$ – nadmorské výšky uvedených bodov (v nejakých vám neznámych jednotkách). Výšky sú uvedené v poradí, v akom na trase nasledujú.

### Formát výstupu

Výstup má obsahovať jediný riadok a v ňom jediné celé číslo $K$ – najväčší počet výšok, ktoré mohli byť vyznačené na Miškovej mape.

Ak $a_1, \ldots, a_K$ sú výšky vyznačené na Miškovej mape, musia spĺňať nasledujúcu podmienku: pre nejaké $x$ z množiny $\{1, \ldots, K\}$ musí platiť:

$$\forall i \in \{1, \ldots, x-1\};\ a_i < a_{i+1} \quad \wedge \quad \forall i \in \{x, \ldots, K-1\};\ a_i > a_{i+1}$$

### Príklad

vstup

```
12
112 247 211 209 244
350 470 510 312 215
117 217
```

výstup

```
9
```

*Jedna možnosť ako mohli vyzerať výšky na Miškovej mape: 112, 211, 244, 350, 470, 510, 312, 215, 117.*

### Pomalšie riešenia

Maximálne 7 bodov: Riešte tú istú úlohu pre $N \leq 1000$.
Maximálne 4 body: Riešte tú istú úlohu pre $N \leq 20$.

## A-I-2 Rezervácie miesteniek

„V horách sa našlo zlato!“ šepkali si ľudia po celej krajine už niekoľko týždňov. Dobrodružnejšie povahy si už balili veci a smerovali priamo do hôr, do legendami opradenej oblasti pýšiacej sa menom Vyšná Klondika.

Keď však Klondiku zaplavila vlna nových zlatokopov, nastal nečakaný problém. Jediný miestny vláčik, ktorý na Klondike mali, odrazu býval tak preplnený, že sa doň polovica záujemcov ani nezmestila. A nedivte sa, že zlatokopa, ktorý sa ženie za čo najvýhodnejšou parcelou, takéto niečo poriadne nazlostí.

Keď to už vyzeralo, že onedlho príde na každej stanici ku krviprelievaniu, dostali železničiari spásonosný nápad. Budú na vláčik predávať miestenky.

### Súťažná úloha

Napíšte program, ktorý bude spracúvať rezervácie miest na jednu jazdu vlaku. Trať vlaku má $N + 1$ staníc, ktoré si v poradí, v akom na trati ležia,

očíslujeme od 0 po $N$. Vo vlaku je $M$ miest, medzi každou dvojicou po sebe nasledujúcich staníc teda vie previezť najviac $M$ zlatokopov.

Váš program bude mať postupne spracovať niekoľko požiadaviek na rezerváciu miest. Každá požiadavka je tvaru „$x$ ľudí chce ísť zo stanice $y$ na stanicu $z$“. Ak ešte je na každom úseku trate medzi stanicami $y$ a $z$ vo vlaku ešte aspoň $x$ miest voľných, má váš program takúto požiadavku prijať, inak ju má odmietnuť.

### Formát vstupu

Prvý riadok vstupu obsahuje tri celé čísla $N$, $M$ a $P$ ($1 \leq N, M, P \leq 100\,000$) – počet úsekov trate, počet miest vo vlaku a počet požiadaviek o rezerváciu.

Nasleduje $P$ riadkov. Každý z nich popisuje jednu požiadavku v poradí, v akom prišli do systému. Presnejšie, $i$-ty z týchto riadkov obsahuje tri celé čísla $x_i$, $y_i$, $z_i$ oddelené medzerami ($1 \leq x_i \leq M$, $0 \leq y_i < z_i \leq N$). Význam týchto hodnôt bol popísaný vyššie.

### Formát výstupu

Pre každú požiadavku vypíšte jeden riadok a v ňom buď reťazec „`prijata`“, ak danú požiadavku bolo ešte možné splniť, alebo reťazec „`odmietnuta`“, ak už vo vlaku nebolo dosť voľných miest.

### Príklad

vstup

```
4 6 6
2 1 4
2 1 3
3 2 4
3 1 2
6 0 1
4 3 4
```

výstup

```
prijata
prijata
odmietnuta
odmietnuta
prijata
prijata
```

*Po prijatí prvých dvoch požiadaviek vieme, že v úsekoch medzi stanicami 1 a 2 a medzi 2 a 3 už budú len dve voľné miesta. Preto musíme odmietnuť*

*nasledujúce dve trojčlenné skupiny, ktoré chcú cestovať aj na týchto úsekoch trate. Posledné dve požiadavky opäť môžeme splniť. U prvej je to zjavné, u druhej nám v stanici 3 vystúpi dosť ľudí na to, aby sa na posledný úsek trate uvoľnili presne tie potrebné štyri miesta.*

### Pomalšie riešenia

Maximálne 7 bodov: Riešte pôvodnú úlohu pre $N \leq 10\,000$, s tým, že navyše môžete predpokladať, že pre približne 99% požiadaviek zo vstupu bude odpoveď „`odmietnuta`“.
Maximálne 4 body: Riešte pôvodnú úlohu pre $N \leq 100$.

## A-I-3 Fibonacciho sústava

Fibonacciho postupnosť vyzerá nasledovne:

$$0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, \ldots$$

Zostrojiť ju vieme tak, že prvé dva členy sú 0 a 1, každý nasledujúci je súčtom dvoch predchádzajúcich. Matematicky zapísané:

$$\begin{aligned} F_0 &= 0 \\ F_1 &= 1 \\ F_{n+2} &= F_{n+1} + F_n \qquad (\forall n \geq 0) \end{aligned}$$

Pozrime sa teraz na to, ako fungujú pozičné číselné sústavy. Pozičná číselná sústava je určená dvoma údajmi: množinou používaných cifier a postupnosťou, ktorá udáva pre každú pozíciu hodnotu, ktorou sa príslušná cifra násobí.

Napríklad naša desiatková sústava používa množinu cifier $\{0, 1, 2, \ldots, 9\}$ a jednotlivé pozície v čísle majú hodnoty (sprava doľava) 1, 10, 100, 1000, ...

Fibonacciho číselná sústava je pozičná číselná sústava, ktorá používa len cifry 0 a 1 a v ktorej sú hodnoty jednotlivých pozícií Fibonacciho čísla (začínajúc od $F_2 = 1$). Teda napr. zápis 10011 vo Fibonacciho sústave predstavuje číslo $1 \cdot 8 + 0 \cdot 5 + 0 \cdot 3 + 1 \cdot 2 + 1 \cdot 1 = 11$.

Na rozdiel od bežných sústav, vo Fibonacciho sústave nemajú niektoré čísla jednoznačný zápis. Napr. aj zápis 10100 predstavuje číslo 11.

## Súťažná úloha

a) *(3 body)* Zápis čísla vo Fibonacciho sústave voláme *pekný*, ak sa v ňom nevyskytujú dve po sebe idúce jednotky. Dokážte, že každé prirodzené číslo má vo Fibonacciho sústave práve jeden pekný zápis. Napíšte program, ktorý zo vstupu načíta prirodzené číslo $N$ a vypíše jeho pekný zápis.

b) *(7 bodov)* Napíšte program, ktorý pre dané $k$, $A$ a $B$ spočíta, koľko čísel z množiny $\{A, A+1, \ldots, B\}$ má v peknom zápise práve $k$ jednotiek.

## Formát vstupu

V podúlohe a) je na vstupe jediné prirodzené číslo $N$ ($1 \leq N \leq 1\,000\,000\,000$).

V podúlohe b) sú na vstupe tri prirodzené čísla $k$, $A$ a $B$ ($1 \leq k \leq 30$, $1 \leq A < B \leq 1\,000\,000\,000$).

## Formát výstupu

V podúlohe a) vypíšte jeden riadok a v ňom reťazec núl a jednotiek, predstavujúci pekný zápis čísla $N$.

V podúlohe b) vypíšte jeden riadok a v ňom jedno celé číslo – počet čísel v zadanom intervale, ktoré majú v peknom zápise práve $k$ jednotiek.

## Príklady pre podúlohu a)

| vstup | výstup |
|---|---|
| 11 | 10100 |

| vstup | výstup |
|---|---|
| 174591 | 10100010000000000100010010 |

### Príklady pre podúlohu b)

| vstup | výstup |
|---|---|
| 1 4 13 | 3 |

Pre $k = 1$ je výstupom počet Fibonacciho čísel v danom rozsahu. V zadanom rozsahu ležia Fibonacciho čísla 5, 8 a 13.

| vstup | výstup |
|---|---|
| 2 4 13 | 6 |

| vstup | výstup |
|---|---|
| 10 102000 103000 | 86 |

### Pomalšie riešenia

Maximálne 2 body za časť b):

Riešte pôvodnú úlohu navyše s podmienkou $|B - A| \leq 100\,000$.

## A-I-4 Prekladacie stroje

V tomto ročníku olympiády sa budeme v teoretickej úlohe stretávať s prekladacími strojmi. V študijnom texte uvedenom za zadaním tejto úlohy sú tieto stroje popísané.

### Súťažná úloha

a) *(1 bod)* Všimnite si prekladacie stroje $B$ a $C$ z príkladov v študijnom texte. Tieto dva stroje zjavne robia preklad „opačnými smermi“. Dalo by sa preto očakávať, že platí nasledujúce tvrdenie:

   Nech $M$ je ľubovoľná (aj nekonečná) množina, ktorej každý prvok je nejaký reťazec písmen $a$, $e$ a $i$. Potom $C(B(M)) = M$.
   Slovne: Keď zoberieme množinu $M$, preložíme ju pomocou $B$ a výsledok preložíme pomocou $C$, dostaneme pôvodnú množinu.

Ak toto tvrdenie naozaj platí, dokážte ho. Ak nie, nájdite protipríklad.

b) *(1 bod)* To isté ako v predchádzajúcej podúlohe, len $M$ obsahuje reťazce tvorené znakmi ▬ a • a zaujíma nás, či musí platiť $B(C(M)) = M$.

c) *(3 body)* Hovoríme, že reťazec je zaujímavý, ak obsahuje len písmená $a$ a $b$, pričom písmen $a$ je dvakrát viac ako písmen $b$. Nech $X$ je množina všetkých zaujímavých reťazcov. Teda napr. $aaabab \in X$, ale $baba \notin X$.

Nech $Y$ je množina všetkých reťazcov, ktoré obsahujú najskôr niekoľko písmen $a$ a za nimi trikrát toľko písmen $b$. Teda napr. $abbb \in Y$, ale $aaabab \notin Y$.

Zostrojte prekladací stroj, ktorý preloží $X$ na $Y$.

d) *(5 bodov)* Na začiatku máme množinu $M_1$, ktorá obsahuje práve všetky reťazce z písmen $a$, $b$, ktoré obsahujú rovnako veľa písmen $a$ ako $b$. Teda napr. $abbbaa \in M_1$, ale $aabab \notin M_1$.

Nové množiny môžeme zostrojovať nasledovnými operáciami:
– prekladom nejakej už zostrojenej množiny nejakým strojom (môžeme použiť zakaždým iný stroj)
– zjednotením dvoch už zostrojených množín
– prienikom dvoch už zostrojených množín

Na čo najmenej operácií zostrojte množinu $G$, ktorá obsahuje práve všetky reťazce, kde je najskôr niekoľko písmen $a$, potom toľko isto $b$, a nakoniec toľko isto $c$. Teda napr. $aabbcc \in G$, ale $abcc \notin G$, a ani $bac \notin G$.

## Študijný text

**Prekladací stroj** je stroj, ktorý dostáva ako vstup reťazec znakov. Tento reťazec postupne číta a podľa vopred zvolenej sady pravidiel (teda svojho programu) občas nejaké znaky zapíše na výstup. Po tom, ako stroj spracuje celý vstupný reťazec a úspešne skončí výpočet, zoberieme reťazec znakov zapísaný na výstup a nazveme ho **prekladom** vstupného reťazca.

Výpočet stroja nemusí byť jednoznačne určený. Inými slovami, sada pravidiel môže niekedy stroju umožniť rozhodnúť sa. V takomto prípade sa môže stať, že niektorý reťazec bude mať viac rôznych prekladov.

Naopak, môže sa stať, že v nejakej situácií sa podľa danej sady pravidiel nebude dať v preklade pokračovať vôbec. V takomto prípade sa môže stať, že niektorý reťazec nebude mať vôbec žiadny preklad.

### Formálnejšia definícia prekladacieho stroja

Každý prekladací stroj pracuje nad nejakou vopred zvolenou konečnou množinou znakov. Túto budeme volať **abeceda** a značiť $\Sigma$. V súťažných úlohách bude vždy $\Sigma$ známa zo zadania úlohy. Abeceda nikdy nebude obsahovať znak \$, ten budeme používať na označenie konca vstupného reťazca.

Stroj si môže počas prekladu reťazca pamätať konečne veľkú informáciu. Formálne toto definujeme tak, že stroj sa v každom okamihu prekladu nachádza v jednom z *konečne veľa* stavov. Nutnou súčasťou programu prekladacieho stroja je teda nejaká konečná **množina stavov**, v ktorých sa môže nachádzať. Túto množinu označíme $K$. Okrem samotnej množiny stavov je taktiež potrebné uviesť, v ktorom stave sa stroj nachádza na začiatku každého prekladu. Tento stav budeme volať **začiatočný stav**.

Program stroja sa bude skladať z konečného počtu prekladových pravidiel. Každé pravidlo je štvorica $(p, u, v, q)$, kde $p, q \in K$ sú nejaké dva stavy a $u, v$ sú nejaké dva reťazce znakov z abecedy $\Sigma$.

Stavy $p$ a $q$ môžu byť aj rovnaké. Reťazec $u$ môže ako svoj jediný znak obsahovať znak \$. Reťazce $u$ a $v$ môžu byť aj rovnaké a môžu byť aj prázdne. Aby sa program ľahšie čítal, budeme namiesto prázdneho reťazca písať $\varepsilon$.

Význam takéhoto pravidla je nasledujúci: „Ak je stroj práve v stave $p$ a neprečítaná časť vstupu začína reťazcom $u$, môže tento reťazec zo vstupu prečítať, na výstup zapísať reťazec $v$ a zmeniť stav na $q$.“ Všimnite si, že pravidlo tvaru $(p, \varepsilon, v, q)$ môžeme použiť vždy, keď sa stroj nachádza v stave $p$, bez ohľadu na to, aké znaky ešte zostávajú na vstupe.

Ešte potrebujeme povedať, kedy preklad úspešne skončil. V prvom rade budeme vyžadovať, aby prekladací stroj dočítal celý vstupný reťazec. Okrem toho umožníme stroju „povedať“, či sa mu preklad podaril alebo nie. Toto spravíme

tak, že niektoré stavy stroja nazveme **ukončovacie stavy**. Množinu ukončovacích stavov budeme značiť $F$.

### Úplne formálna definícia prekladacieho stroja

Prekladací stroj je usporiadaná pätica $(K, \Sigma, P, q_0, F)$, kde $\Sigma$ a $K$ sú *konečné* množiny, $q_0 \in K$, $F \subseteq K$ a $P$ je *konečná* množina prekladových pravidiel popísaných vyššie. Presnejšie, nech $\Sigma^*$ je množina všetkých reťazcov tvorených znakmi zo $\Sigma$, potom $P$ je *konečná* podmnožina množiny $K \times (\Sigma^* \cup \{\$\}) \times \Sigma^* \times K$.

(Pre každé $q \in K$ budeme množinu pravidiel, ktorých prvá zložka je $q$, volať „prekladové pravidlá zo stavu $q$“.)

Ak teda chceme definovať konkrétny prekladací stroj, musíme uviesť všetkých päť vyššie uvedených objektov.

Keď už máme definovaný konkrétny stroj $A = (K, \Sigma, P, q_0, F)$, môžeme definovať, ako tento stroj prekladá konkrétny reťazec. Povieme najskôr formálnu definíciu, potom ju slovne vysvetlíme.

Množina **platných prekladov** reťazca $u$ prekladacím strojom $A$ je:

$$A(u) = \Big\{ \quad v \quad \Big| \quad \begin{array}{l} \exists n \geq 0; \quad \exists (p_1, u_1, v_1, r_1), \ldots, (p_n, u_n, v_n, r_n) \in P; \\ (\forall i \in \{1, \ldots, n-1\};\ r_i = p_{i+1}) \ \wedge \ p_1 = q_0 \ \wedge \ r_n \in F \ \wedge \\ \wedge\ \exists k \geq 0;\ u_1 u_2 \ldots u_n = u \underbrace{\$ \ldots \$}_{k} \ \wedge \ v_1 v_2 \ldots v_n = v \end{array} \Big\}$$

Definícia hovorí, kedy je reťazec $v$ prekladom reťazca $u$. Vysvetlíme slovne význam jednotlivých riadkov definície:

- Prvý riadok hovorí, že aby sa dalo $u$ preložiť na $v$, musí existovať nejaká postupnosť prekladových pravidiel, ktorú pri tomto preklade použijeme. Zvyšné dva riadky popisujú, ako táto postupnosť musí vyzerať.
- Druhý riadok zabezpečuje, aby stavy v použitých pravidlách boli správne: Prvé pravidlo musí byť pravidlom zo začiatočného stavu, každé ďalšie pravidlo musí byť pravidlom z toho stavu, do ktorého sa stroj dostal použitím predchádzajúceho pravidla.

Navyše stav, v ktorom bude stroj na konci výpočtu, musí byť ukončovací.

- Posledný riadok hovorí o reťazcoch, ktoré stroj pri použití dotyčných prekladových pravidiel prečíta a zapíše.

  Reťazec, ktorý pri použití týchto pravidiel stroj prečíta zo vstupu, musí byť naozaj zadaný reťazec $u$, prípadne sprava doplnený vhodným počtom znakov \$.

  Reťazec, ktorý stroj zapíše na výstup, musí byť presne reťazec $v$.

### Na čo budeme používať prekladacie stroje?

Nám budú prekladacie stroje slúžiť na preklad jednej množiny reťazcov na inú množinu reťazcov. Ak $A$ je prekladací stroj a $M \subseteq \Sigma^*$ nejaká množina reťazcov, tak preklad množiny $M$ strojom $A$ je množina $A(M) = \bigcup_{u \in M} A(u)$.

Inými slovami, výslednú množinu $A(M)$ zostrojíme tak, že zoberieme všetky reťazce z $M$ a pre každé z nich dáme do $A(M)$ všetky jeho platné preklady.

### Príklad 1

Majme abecedu $\Sigma = \{0, \ldots, 9\}$. Nech $M$ je množina všetkých tých reťazcov, ktoré predstavujú zápisy kladných celých čísel v desiatkovej sústave. Zostrojíme prekladací stroj $A$, pre ktorý bude platiť, že prekladom tejto množiny $M$ bude množina zápisov všetkých kladných celých čísel, ktoré sú deliteľné tromi.

### Riešenie

Najjednoduchšie bude jednoducho vybrať z $M$ tie čísla, ktoré sú deliteľné tromi. Náš prekladací stroj bude kopírovať cifry zo vstupu na výstup, pričom si bude v stave pamätať, aký zvyšok po delení tromi dáva doteraz prečítané (a zapísané) číslo. Ak je po dočítaní vstupu v stave zodpovedajúcom zvyšku 0, prejde do ukončovacieho stavu.

Formálne $A$ bude pätica $(K, \Sigma, P, 0, F)$, kde $K = \{0, 1, 2, end\}$, $F = \{end\}$ a prekladové pravidlá vyzerajú nasledovne:

$$P = \Big\{(x, y, y, z) \;\Big|\; x \in \{0, 1, 2\} \wedge y \in \Sigma \wedge z = (10x{+}y) \bmod 3\Big\} \cup \Big\{(0, \$, \varepsilon, end)\Big\}$$

**Príklad 2**

Majme abecedu $\Sigma = \{a, e, i, \bullet, \text{—}\}$. Zostrojíme prekladací stroj $B$, pre ktorý bude platiť, že prekladom ľubovoľnej množiny $M$, ktorá obsahuje len reťazce z písmen $a$, $e$ a $i$, bude množina tých istých reťazcov v morzeovke (bez oddeľovačov medzi znakmi). Zápisy našich písmen v morzeovke sú nasledujúce: $a$ je $\bullet$ —, $e$ je $\bullet$ a $i$ je $\bullet\bullet$.

Napríklad množinu $M = \{ae, eea, ia\}$ by náš stroj mal preložiť na množinu $\{\bullet$ — $\bullet, \bullet\ \bullet\ \bullet$ —$\}$. (Všimnite si, že reťazce $eea$ a $ia$ majú v morzeovke bez oddeľovačov rovnaký zápis.)

**Riešenie**

Prekladací stroj $B$ bude jednoducho čítať vstupný reťazec po znakoch a zakaždým zapíše na výstup kód prečítaného znaku.

Formálne $B$ bude pätica $(K, \Sigma, P, \heartsuit, F)$, kde $K = \{\heartsuit\}$, $F = \{\heartsuit\}$ a prekladové pravidlá vyzerajú nasledovne:

$$P = \{(\heartsuit, a, \bullet\ \text{—}, \heartsuit),\ (\heartsuit, e, \bullet, \heartsuit),\ (\heartsuit, i, \bullet\bullet, \heartsuit)\}$$

Všimnite si, že nepotrebujeme explicitne kontrolovať, či sme na konci vstupu. Totiž počas celého prekladu sme v ukončovacom stave, a teda akonáhle prečítame posledný znak zo vstupu, je zapísaný preklad platný.

**Príklad 3**

Majme abecedu $\Sigma = \{a, e, i, \bullet, \text{—}\}$. Zostrojíme prekladací stroj $C$, pre ktorý bude platiť, že prekladom ľubovoľnej množiny $M$, ktorá obsahuje len reťazce zo znakov $\bullet$ a —, bude množina **všetkých** reťazcov z písmen $a$, $e$ a $i$, ktoré keď zapíšeme v morzeovke (bez oddeľovačov medzi znakmi), dostaneme reťazec z $M$. Napr. množinu $M = \{\bullet$ — $\bullet, \bullet\ \bullet\ \bullet$ —$\}$ by náš stroj mal preložiť na $\{ae, eea, ia\}$.

**Riešenie**

Nášmu prekladaciemu stroju dáme možnosť sa v každom okamihu prekladu rozhodnúť, že ide čítať kód nejakého písmena a zapísať na výstup toto písmeno. Potom každej možnosti, ako rozdeliť vstupný reťazec na kódy písmen, bude zodpovedať jeden platný preklad.

Formálne $C$ bude pätica $(K, \Sigma, P, \diamondsuit, F)$, kde $K = \{\diamondsuit\}$, $F = \{\diamondsuit\}$ a prekladové pravidlá $P = \{(\diamondsuit, \bullet\ \text{—}, a, \diamondsuit),\ (\diamondsuit, \bullet, e, \diamondsuit),\ (\diamondsuit, \bullet\bullet, i, \diamondsuit)\}$.

Ako príklad si ukážeme, ako mohol prebiehať preklad reťazcov z vyššie uvedenej množiny $M$. Sú tieto tri možnosti:
$(\diamondsuit, \bullet\!\!-, a, \diamondsuit)$, $(\diamondsuit, \bullet, e, \diamondsuit)$
$(\diamondsuit, \bullet\bullet, i, \diamondsuit)$, $(\diamondsuit, \bullet\!\!-, a, \diamondsuit)$
$(\diamondsuit, \bullet, e, \diamondsuit)$, $(\diamondsuit, \bullet, e, \diamondsuit)$, $(\diamondsuit, \bullet\!\!-, a, \diamondsuit)$
Ak by sme skúsili na ľubovoľnom vstupe z $M$ používať prekladové pravidlá v inom poradí – napr. na vstupe $\bullet\bullet\bullet\!\!-$ použiť trikrát pravidlo $(\diamondsuit, \bullet, e, \diamondsuit)$ – nepodarí sa nám dočítať vstupný reťazec do konca.

## Príklad 4

Majme abecedu $\Sigma = \{a, b, c\}$. Nech množina $X$ obsahuje práve všetky tie reťazce, v ktorých je rovnako $a$ a $b$. Teda napr. $abbccac \in X$, ale $cbaa \notin X$.

Nech množina $Y$ obsahuje práve všetky tie reťazce, ktoré neobsahujú žiadne $a$, neobsahujú podreťazec $bc$, a písmen $c$ je dvakrát toľko ako písmen $b$. Teda napr. $ccccbb \in Y$, ale $cccbcb \notin Y$ a $acacba \notin Y$.

Zostrojíme prekladací stroj $D$, ktorý preloží $X$ na $Y$.

## Riešenie

Budeme prekladať len niektoré vhodné reťazce z množiny $X$, presnejšie to budú tie, v ktorých nie sú žiadne $c$ a všetky $a$ idú pred všetkými $b$. Takýto reťazec preložíme tak, že najskôr každé $a$ prepíšeme na $cc$, a potom skopírujeme na výstup všetky $b$. Teda napr. prekladom slova $aabb$ bude slovo $ccccbb$.

Formálne $D$ bude pätica $(K, \Sigma, P, \textbf{čítaj-a}, F)$, kde $K = \{\textbf{čítaj-a}, \textbf{čítaj-b}\}$, $F = \{\textbf{čítaj-b}\}$ a prekladové pravidlá vyzerajú nasledovne:
$P = \{(\textbf{čítaj-a}, a, cc, \textbf{čítaj-a}),\ (\textbf{čítaj-a}, b, b, \textbf{čítaj-b}),\ (\textbf{čítaj-b}, b, b, \textbf{čítaj-b})\}$

Prečo tento prekladací stroj funguje? Ak vstupný reťazec obsahuje nejaké písmeno $c$, pri jeho prekladaní sa pri prvom výskyte $c$ náš stroj zasekne. Preto takéto reťazce nemajú žiaden platný preklad. Podobne nemajú platný preklad reťazce, kde nie je dodržané poradie písmen $a$ a $b$. Totiž akonáhle náš stroj prečíta prvé $b$, prejde do stavu **čítaj-b**, a ak sa teraz ešte na vstupe vyskytne $a$, stroj sa zasekne.

Platné preklady teda existujú naozaj len pre slová vyššie popísaného tvaru. Je zjavné, že z každého z nich vyrobíme nejaký reťazec z $Y$, preto $D(X) \subseteq Y$. A naopak, ak si povieme konkrétne slovo z $Y$, ľahko nájdeme slovo z $X$, ktoré sa naň preloží, preto $Y \subseteq D(X)$, a teda $Y = D(X)$.

---

**Adresy krajských komisií OI**

*Banskobystrický kraj*
PaedDr. Miloslava Sudolská, KI FPV UMB, Tajovského 40, 974 01 B. Bystrica

*Bratislavský kraj*
RNDr. Eva Hanulová, Gym. J. Hronca, Novohradská 3, 821 09 Bratislava.

*Košický kraj*
RNDr. Rastislav Krivoš-Belluš, Ústav informatiky PF UPJŠ,
Jesenná 5, 041 54 Košice

*Nitriansky kraj*
prof. Ing. Veronika Stoffová, CSc., KI PF UJS, Roľníckej školy 1519,
945 01 Komárno

*Prešovský kraj*
Mgr. Mária Majherová, Katedra matematiky, FHPV, Prešovská univerzita,
Ul. 17 novembra č. 1, 081 16 Prešov

*Trenčiansky kraj*
Ing. Andrea Julény, Katedra informatiky, Fakulta mechatroniky TnUAD,
Pri parku 19, 911 06 Trenčín

*Trnavský kraj*
Mgr. Blanka Thomková, Gym. Jána Hollého, Na hlinách 30, 917 01 Trnava

*Žilinský kraj*
RNDr. Peter Varša, Ph.D., KI FRI ŽU, Moyzesova 20, 010 26 Žilina

---

SLOVENSKÁ KOMISIA OLYMPIÁDY V INFORMATIKE

**DVADSIATY TRETÍ ROČNÍK OLYMPIÁDY V INFORMATIKE**

Vydala IUVENTA s finančnou podporou Ministerstva školstva SR
Náklad: 400 výtlačkov
Zodpovedný redaktor: Michal Forišek
Sadzba programom LaTeX