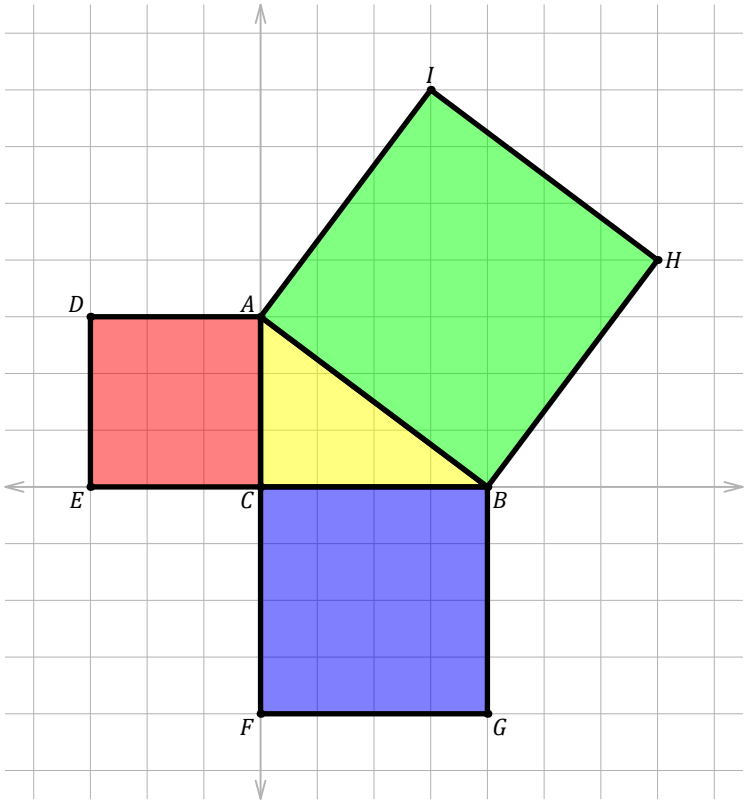


Informatika v prírodných vedách a matematike
modul Matematika

EURÓPSKA ÚNIA
Európsky sociálny fond
Európsky fond regionálneho rozvoja

OPERAČNÝ PROGRAM
ĽUDSKÉ ZDROJE

MINISTERSTVO
ŠKOLSTVA, VEDY,
VÝSKUMU A ŠPORTU
SLOVENSKEJ REPUBLIKY

it akadémia

Tento projekt sa realizuje vďaka podpore z Európskeho sociálneho fondu
v rámci Operačného programu Ľudské zdroje

www.minedu.sk www.employment.gov.sk/sk/esf/ www.itakademia.sk

Informatika v prírodných vedách a matematike
modul Matematika

Komentovaný učebný text

Spracované v rámci národného projektu
IT Akadémia – vzdelávanie pre 21. storočie

Bratislava 2021

Informatika v prírodných vedách a matematike – modul Matematika

Komentovaný učebný text

Spracované v rámci národného projektu IT Akadémia – vzdelávanie pre 21. storočie

Autor: prof. RNDr. Stanislav Krajči, PhD.

Recenzenti: Ing. Božena Mannová, M. Math. Ph. D., Ing. Juraj Ťapák

Neprešlo jazykovou úpravou

Vydavateľ: Centrum vedecko-technických informácií SR, Bratislava

Rok vydania: 2021

Vydanie: 2.

ISBN: 978-80-89965-70-0

Bratislava 2020

Tento projekt sa realizuje vďaka podpore z Európskeho sociálneho fondu v rámci Operačného programu Ľudské zdroje

Obsah

Úvodom

Keď sa pozrieme na obsah stredoškolskej matematiky, s prekvapením zistíme, že oblastí, kde sa používajú (naprogramovateľné) algoritmy, nie je príliš veľa. Asi najnáročnejším postupom stredoškolskej matematiky je úprava algebraických výrazov, avšak jej prepis do exaktného algoritmu je nesmierne náročný. Často sa pri výpočtoch používajú vzorce (najzložitejší je azda vzorec pre výpočet koreňov kvadratickej rovnice), tie však majú (aj z programátorského pohľadu) jednoriadkový charakter. Za náročnejší možno považovať rekurzívny Euklidov algoritmus na nájdenie najväčšieho spoločného deliteľa dvoch prirodzených čísel či Gaussovu eliminačnú metódu na riešenie sústav lineárnych rovníc. Tu si treba uvedomiť, že použitie softvéru na riešenie takýchto a podobných problémov je na strednej škole dosť kontraproduktívne, pretože pri nich zreteľne nejde o samotný výsledok, ale o porozumenie postupu, ktorý k nemu vedie. Použitie programu by tak zatienilo práve tú časť riešeného problému, o ktorý pri výučbe ide. Softvér sa môže použiť až vtedy, keď žiaci metódu úplne pochopili a jej ďalšie manuálne používanie ich už iba zdržiava. Platí skrátka to isté, čo v nižších ročníkoch pre kalkulačku.

Ďalšou didaktickou nevýhodou všetkých týchto algoritmov je, že nie sú variovateľné. Stačí ich raz správne naprogramovať, a potom ich už možno bez akejkoľvek zmeny iba používať. Žiaden priestor pre tvorivosť tam jednoducho neostáva.

Existuje však oblasť stredoškolskej matematiky, ktorá touto vlastnosťou netrpí, a to sú geometrické konštrukčné úlohy. Nejeden stredoškolský žiak berie zápis postupu konštrukcie do jednotlivých bodov ako nepríjemnú povinnosť, ktorej zmysel príliš nechápe. Zápisy konštrukcií však možno tvorivo didakticky využiť: Dvojica žiakov si zahrá takú hru, že jeden žiak popíše svoju konštrukciu a jeho sused správnosť tohto popisu skontroluje tým, že podľa neho obrázok spätne narysuje a výsledok porovná s originálom. Zápis konštrukcie tak môžeme chápať ako (pomerne jednoduchý, lebo sekvenčný) program (hoci nie práve v klasickom programovacom jazyku), ktorého inštrukciami sú viac či menej elementárne konštrukcie („narysuj priamku“, „označ priesečník“, „zostroj kružnicu“, prípadne „zostroj stred úsečky“ a podobne), ktoré umožňujú veľkú variabilitu. Táto oblasť matematiky má teda výrazný programátorský potenciál.

Aj tu platí, že žiaci musia pri konštrukciách prejsť najprv manuálnou fázou, aby sa ich zručnosti mali dostatočný čas utvrdiť. Po istom čase (a návratov ku konštrukčným úlohám je počas štyroch stredoškolských rokov viacero) však začne byť samotné rysovanie otravné a zdržujúce a stráca sa tak čas, ktorý by mohol byť využitý efektívnejšie, na menej rutinné veci. Vzniká preto prirodzená potreba samotné konštrukcie urýchliť, zrejme nejakým vhodným softvérom. Azda najznámejšími sú programy Cabri a GeoGebra, no na naše ciele bude vhodnejší Asymptote, ktorý v spolupráci s typografickým systémom TeX produkuje (na rozdiel od spomínaných dvoch) profesionálny výstup vo formáte .pdf (a podobne ako GeoGebra je zadarmo). Asymptote je plnohodnotný programovací jazyk, nám však postačí pár jeho sekvenčných príkazov (ktoré viac-menej zodpovedajú jednotlivým bodom zápisu konštrukcie), komplikovanejšie programátorské konštrukty, ako sú vetvenie alebo cyklus, tu vôbec netreba. Za pár hodín tak možno zvládnuť aj tie najzložitejšie konštrukčné úlohy. Ako sa hovorí, za málo peňazí veľa muziky ;-).

Tento predmet sleduje viacero didaktických cieľov, a to ako informatických, tak matematických:

- Upozorňuje žiakov, že programovania sa netreba báť, pretože ho v istej forme už dávno poznajú.
- Veľmi rýchlo poskytuje doslova viditeľný výsledok práce (navyše s profesionálnym vzhľadom), čo značne pomáha motivácii.
- Umožňuje okamžite pozorovať, ako sa po zmene nejakého parametra zmení výsledok.
- Ukazuje základný princíp (akéhokoľvek) programovacieho jazyka, a to, že jeho programy používajú niekoľko málo elementárnych inštrukcií, ktoré možno kombinovať do zložitejších celkov.
- Prirodzeným spôsobom učí žiakov používať premenné a dátové typy, prípadne i ďalšie črty jazyka.
- Pracuje s explicitným programovým kódom s pomerne štandardnou syntaxou.
- Pomáha zdôvodňovať zmysluplnosť zápisov konštrukcií.
- Cibrí v žiakoch schopnosť exaktného vyjadrovania (inak predsa nevznikne príslušný obrázok).
- Popri samotných konštrukčných úlohách prehlbuje poznatky z príbuzných oblastí matematiky (napríklad z práce s vektormi).

Metodické usmernenia

Priebeh výučby

Výučba prebieha v počítačovej učebni, kde má každý žiak osobitnú pracovnú stanicu a učiteľova pracovná plocha je zobrazovaná dataprojektorom.

Pred úvodnou hodinou si učiteľ na svojom počítači pripraví výučbové prostredie, žiaci tak urobia na svojich počítačoch počas nej. Celá výučba potom spočíva v postupnom riešení úloh z tohto učebného textu, keď každý (včítane učiteľa) na svojom počítači zapisuje zodpovedajúci programový kód.

Príklady riešia všetci spoločne, v prípade potreby učiteľ žiakov navádza na správne riešenie, pričom striedavo premieta časti tohto učebného textu a svoje výučbové prostredie. Niektoré príklady sú nasledované úlohami, ktoré na nich nadväzujú a ktoré by žiaci mali bez ťažkostí zvládnuť aj bez pomoci učiteľa.

Výskumné úlohy sú určené na samostatné riešenie, čo však neznamená, že žiakom treba brániť vo vzájomnej aktívnej spolupráci. Cieľom je obvykle vyslovenie rozumnej hypotézy o vzťahoch niektorých objektov zo zadania. Po uplynutí primeraného času učiteľ vyzve niektorého žiaka, aby vyslovil svoju hypotézu, a ostatní si potom samostatne preveria jej pravdivosť. Prácu na výskumných úlohách považujeme za najdôležitejšiu časť tohto predmetu.

Vzhľadom na časovú náročnosť navrhovaného učiva odporúčame nezdržiavať žiakov povinnosťou písať si poznámky na papier. Z rovnakých dôvodov môže učiteľ žiakom sprístupniť na skopírovanie niektoré hotové programy alebo ich časti. Takto ušetrený čas nech radšej venujú ďalším experimentom so svojím programom, napríklad pozorovaniam, ako vplýva zmena vstupných parametrov na výsledok.

Výučbové prostredie

Na výučbu potrebujeme hlavne webovské prostredie *Overleaf*, jeho príprava je detailne popísaná v dokumente https://ics.upjs.sk/~krajci/skola/ine/ITAkademia/InformatikaVMatematike/Overleaf/pripravaProstredia.html.

Tento softvér síce primárne slúži na prácu s univerzálnym typografickým systémom TEX, ale dokáže obslúžiť aj programy v jazyku Asymptote. Netreba zaň platiť a neobťažuje ani reklamami.

Zdôraznime, že zaregistrovaný používateľ má k svojim projektom prístup z ľubovoľného miesta internetu, žiaci tak prípadne môžu vo svojej práci pokračovať aj doma.

Potrebné súbory

Zo stránky https://ics.upjs.sk/~krajci/skola/ine/ITAkademia/InformatikaVMatematike/subory/ si treba stiahnuť tieto súbory:

- `konstrukcie.tex` – TEX-ovská obálka, ktorá volá ďalšie súbory
- `uloha.tex` – miesto, kde sa píše program
- `asydef2.tex` – knižnica na prácu v rovine, používaná v kapitolách 1, 2 a 3
- `asydef3.tex` – knižnica na prácu v priestore, používaná v kapitole 4
- `riesenia.zip` – programy (aj prípadné čiastkové) zo všetkých príkladov, úloh a výskumných úloh

Súbory `asydef2.tex` a `asydef3.tex` sa v priebehu výučby (na príslušné výzvy v učebnom texte) dopĺňajú o vlastné príkazy.

Pred začiatkom kapitoly 4 treba riadky

```
\input{asydef2}
%\input{asydef3}
```

súboru `konstrukcie.tex` upraviť takto:

```
%\input{asydef2}
\input{asydef3}
```

Odstraňovanie chýb v programoch

Prostredie *Overleaf* sa pri chybe v programe v jazyku Asymptote správa dosť zákerne: nielenže na ňu poriadne neupozorní, ale nezriedka zobrazí obrázok vytvorený predchádzajúcou verziou programu. Býva preto užitočné pri ladení programu urobiť aspoň minimálnu zmenu, ktorú možno vo výsledku ľahko zaregistrovať (napríklad pozmeniť farbu niektorého objektu), aby sme videli, či sa naša úprava naozaj prejaví.

Menej skúseným programátorom preto pri vytváraní programu odporúčame priebežnou kompiláciou kontrolovať, či sa výsledok vyvíja očakávaným smerom. Užitočnou pomôckou pri hľadaní chyby býva dočasné znefunkčnenie niektorých riadkov programu (napríklad vložením dvoch lomiek (`//`) pred každý z nich).

Drvivá väčšina chýb vzniká porušovaním týchto pravidiel:

- Program sa nesmie začínať prázdnym riadkom. Inými slovami, vzápätí po uvádzajúcom riadku

  ```
  \begin{asy}
  ```

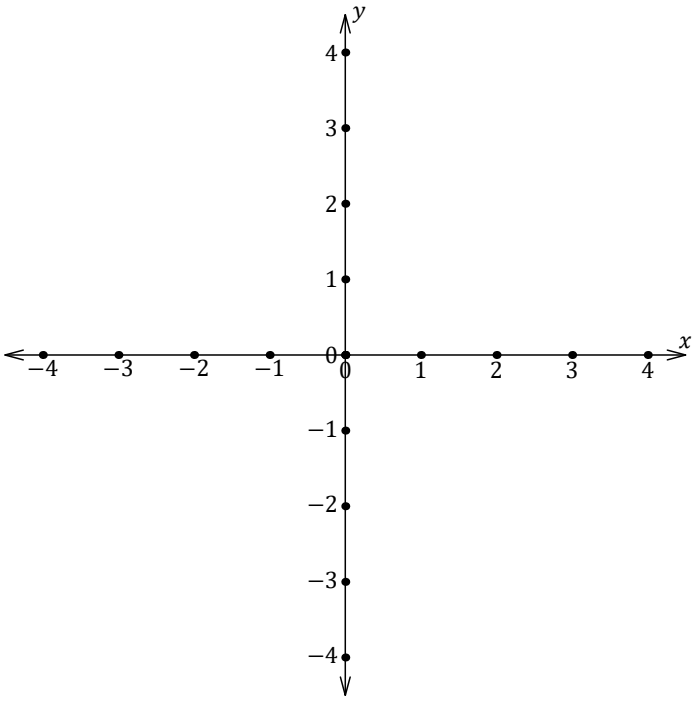
  musí nasledovať prvý riadok programu.
- Každý príkaz sa musí končiť bodkočiarkou.
- V kľúčových slovách (a to aj vlastných) sa nesmú používať písmená s diakritikou.
- Pri deklarácii premennej jej treba určiť správny typ (občas sa mýlia `pair` a `path`).

1

Body a úsečky

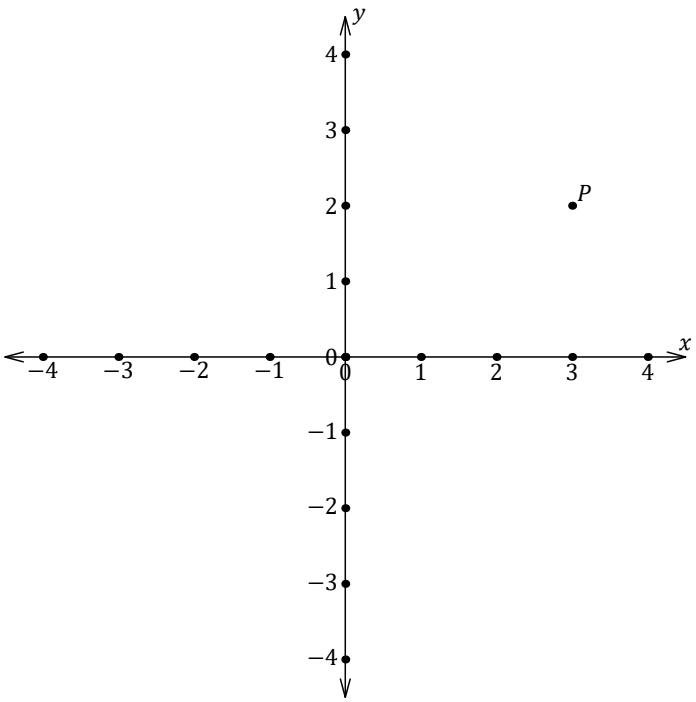
1.1 Body

Pod *súradnicovou sústavou* v rovine obvykle rozumieme dve navzájom kolmé číselné osi, jednu vodorovnú, nazývanú aj *os x*, a druhú zvislú, *os y*, ktoré majú spoločný počiatok:

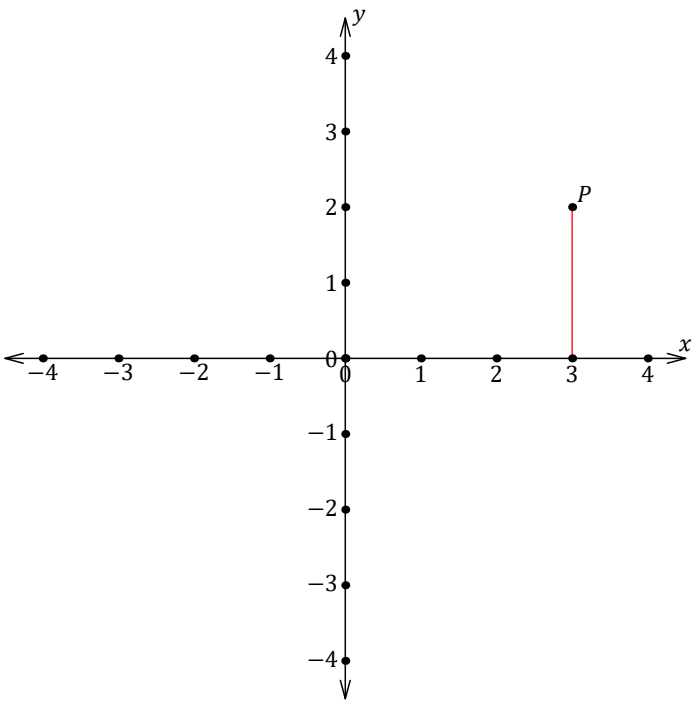


Každý bod roviny potom môžeme jednoznačne popísať tak, že z neho vedieme kolmice na tieto dve osi a priradíme mu čísla z oboch vzniknutých priesečníkov. Tieto dve čísla nazývame jeho *súradnice*. Tá z osi x sa bude volať *x-ová* alebo *prvá* a tá z osi y bude *y-ová* alebo aj *druhá*. Bod s x-ovou súradnicou a a y-ovou súradnicou b tak môžeme stotožniť s dvojicou čísel (a, b).

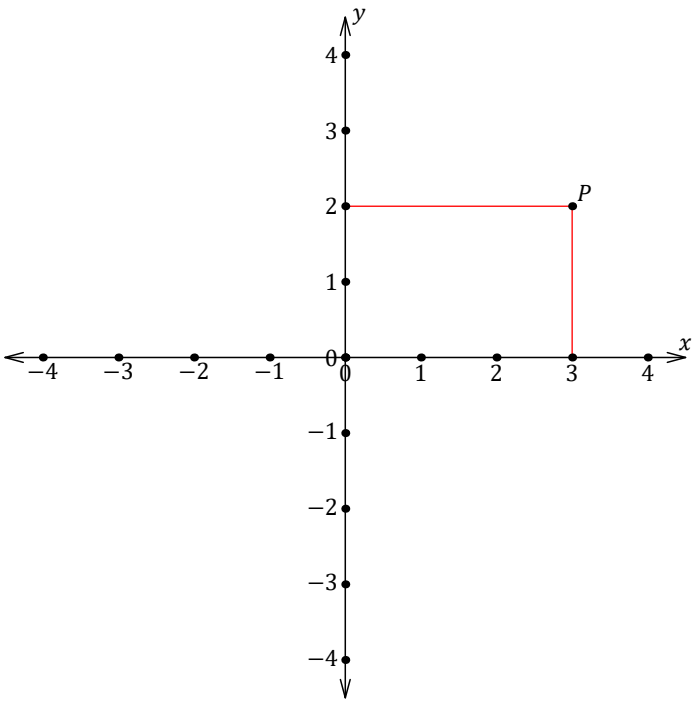
Majme napríklad nasledujúci bod P:



Veďme z neho kolmicu na vodorovnú os. Tá ju pretína v jej bode označujúcom číslo 3, čo znamená, že x-ová súradnica bodu P bude 3:

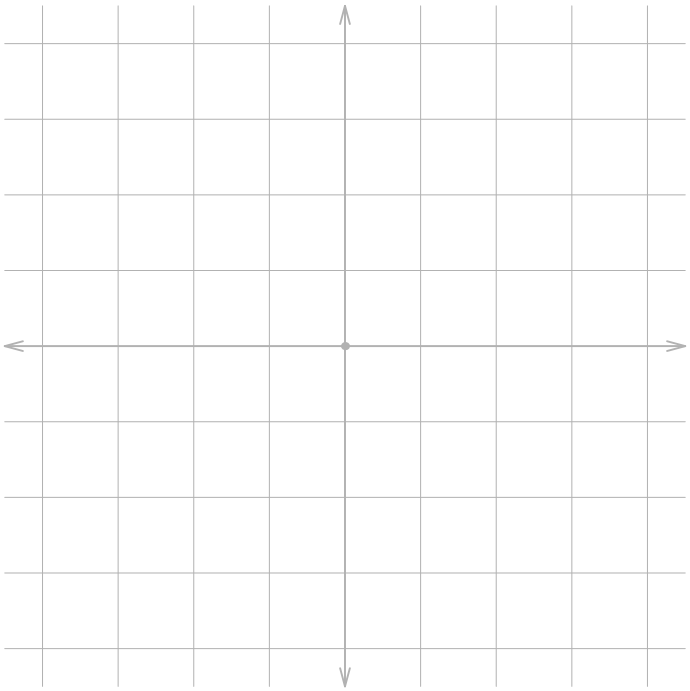


Teraz z bodu P veďme kolmicu na druhú, zvislú os. Ich priesečník označuje číslo 2 – to bude y-ová súradnica nášho bodu.

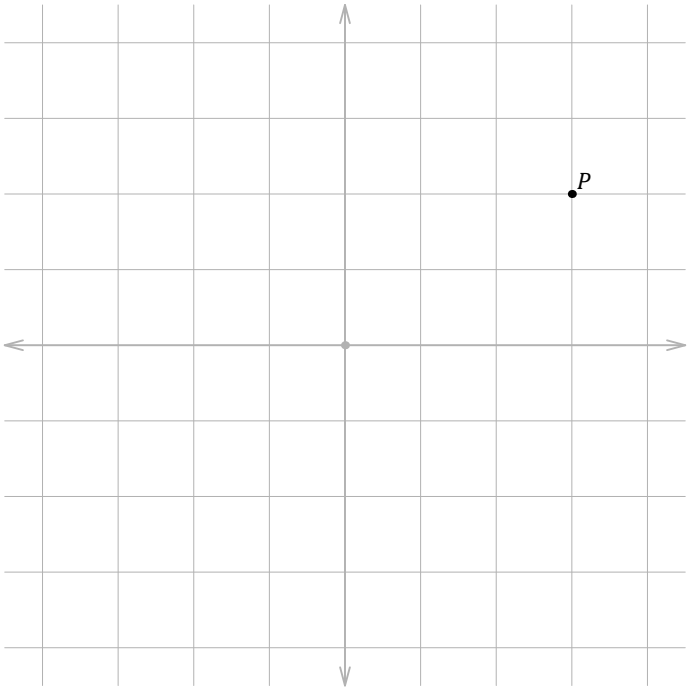


Zhrnutím dostávame, že $P = (3, 2)$.

Kvôli lepšej orientácii budeme našu súradnicovú sústavu znázorňovať spolu s rovnobežkami s osami, ktoré prechádzajú ich celými bodmi. Naopak, označenia týchto čísel na osiach i pomenovania osí vynecháme. Keďže táto štvorčeková sieť bude len podkladom pre naše ďalšie konštrukcie, utlmíme aj jej farbu:

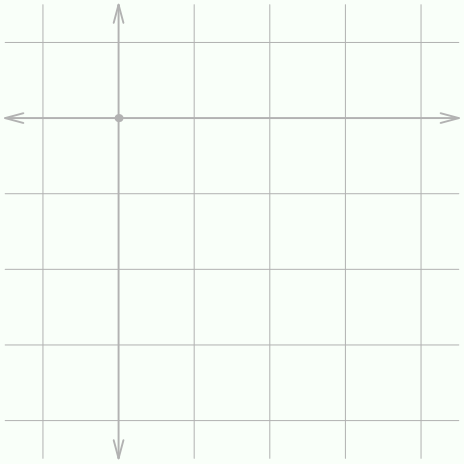


Vrcholy vzniknutých štvorčekov sa nazývajú *mrežové body* a sú to vlastne body, ktorých obe súradnice sú celé čísla. Jedným z nich je i náš bod P, ktorý v tejto sieti vyzerá takto:



Príklad 1:

Nakreslite takúto časť štvorčekovej siete:



Riešenie

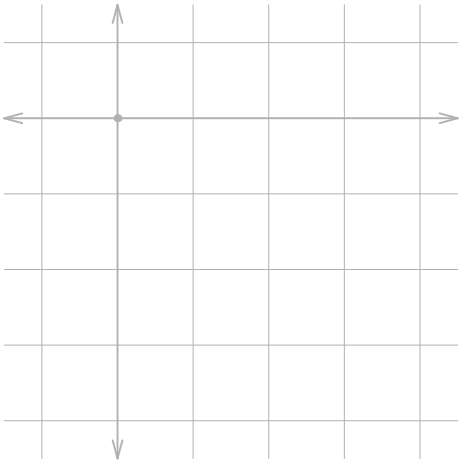
- Súradnicovú štvorčekovú sieť vykreslíme pomocou príkazu `kresliSiet`.
  (Slovenčina (i keď bez diakritiky) použitá v jeho názve naznačuje, že nejde o pôvodný príkaz jazyka Asymptote, ale o naše makro, ktorého zdrojový kód možno nájsť v knižnici `asydef2.tex`.)
- Predvolený rozsah vykreslenej časti siete (v oboch smeroch od -4 po 4) možno zmeniť tak, že medzi zátvorky v príkaze `kresliSiet()` napíšeme čiarkami oddelené všetky štyri požadované hranice, a to v poradí ľavá, pravá, dolná a horná.

Každý príkaz sa končí bodkočiarkou.

Ako vidieť na obrázku zo zadania, naša sieť má vo vodorovnom smere ľavú hranicu -1 a pravú 4 a vo zvislom dolnú hranicu -4 a pravú 1. Tieto štyri čísla teda uvedieme v takomto poradí ako vstupy príkazu `kresliSiet`, takže náš program bude tvorený týmto jediným riadkom:

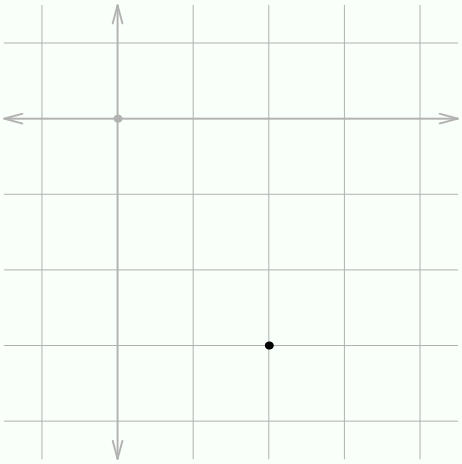
```
kresliSiet(-1,4, -4,1);
```

Dostávame tak požadovaný obrázok:



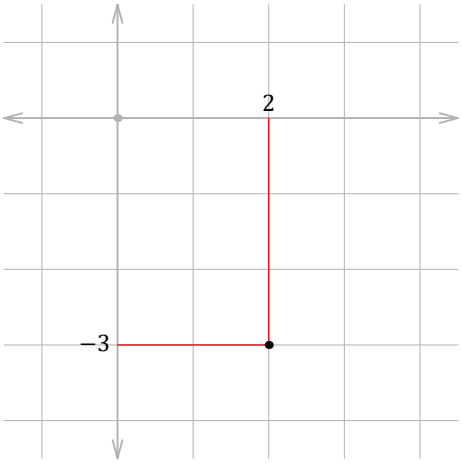
Príklad 2:

V sieti z predchádzajúcej úlohy znázornite tento bod:



Riešenie

Z obrázku zistíme (podľa vyššie uvedeného postupu) obe súradnice bodu:



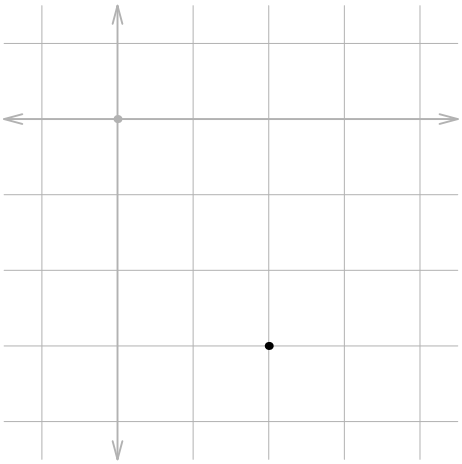
Náš bod je teda $(2, -3)$. Treba ho ešte nejako zapracovať do programu.

- Bod (a, b) budeme zapisovať v tvare `(a,b)`, pričom a a b sú napísané príslušne zmeneným fontom. Čiarku oddeľujúcu obe súradnice nemožno zameniť za bodkočiarku ani za žiaden iný znak.
- Na nakreslenie bodu sa používa príkaz `dot` (bodka), ktorý je nasledovaný bodom v okrúhlych zátvorkách.

Náš bod teda vykreslíme pomocou príkazu `dot((2,-3))`. (Všimnime si, že okolo dvojíc súradníc bodu v druhom riadku sú dva páry zátvoriek – vnútorné z dvoch čísel vytvárajú ich dvojicu, vonkajšie sú súčasťou príkazu `dot`.)
Celý program teda vznikne doplnením tohto príkazu do programu z predchádzajúcej úlohy.

```
kresliSiet(-1,4, -4,1);

dot((2,-3));
```

Jeho výsledok sa zhoduje s obrázkom zo zadania:



Výskumná úloha 1:

Zistite, ako sa zmení poloha zobrazeného bodu, keď jeho:
a) prvú súradnicu zmenšíme;
b) prvú súradnicu zväčšíme;
c) druhú súradnicu zmenšíme;
d) druhú súradnicu zväčšíme.

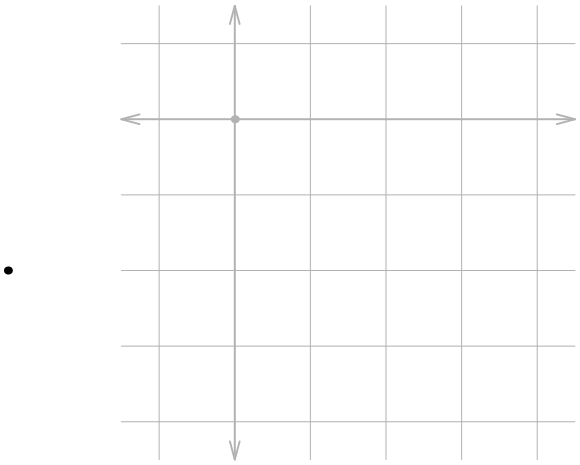
Príklad 3:

Nakreslite bod $(-3, -2)$.

Riešenie

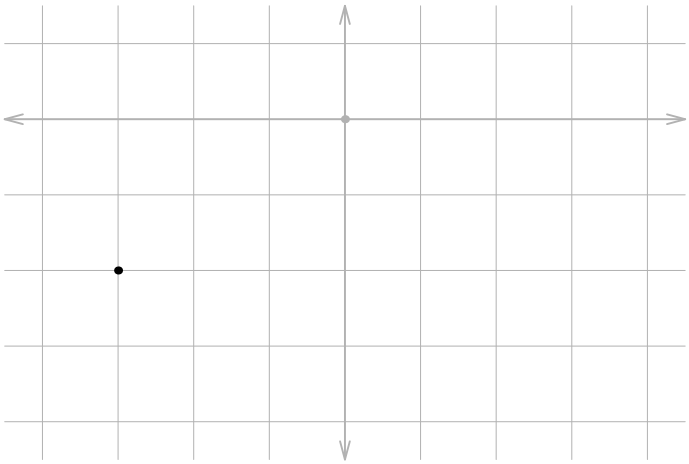
Oproti predchádzajúcemu príkladu máme situáciu zjednodušenú tým, že súradnice zobrazovaného bodu nemusíme zisťovať, lebo sú explicitne dané. Núka sa preto jednoduchá úprava predchádzajúceho riešenia:

```
kresliSiet(-1,4, -4,1);

dot((-3,-2));
```



Ako však vidieť, zobrazený bod sa ocitol mimo vykreslenej časti siete, takže tá tým stráca svoj význam. Preto upravíme aj ju:

```
kresliSiet(-4,4, -4,1);

dot((-3,-2));
```



Príklad 4:

Nakreslite bod $(-2{,}2, 1{,}5)$.

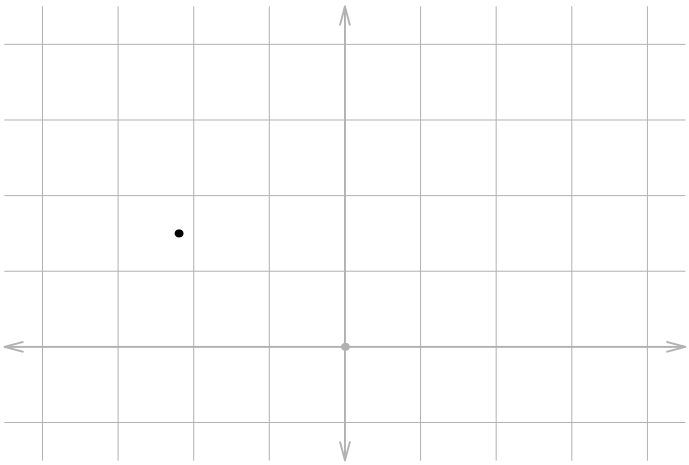
Riešenie

Jedinou zmenou tu je, že tento bod nie je mrežový.

- Namiesto desatinnej čiarky sa píše desatinná bodka.
- Kvôli prehľadnosti možno (napríklad po čiarke oddeľujúce súradnice) pridať medzeru.

Keďže zadanie tejto úlohy sa nápadne podobá na zadanie predchádzajúcej, stačí v programe, ktorý je jej riešením, príslušne upraviť údaje:

```
kresliSiet(-4,4, -1,4);

dot((-2.2,1.5));
```



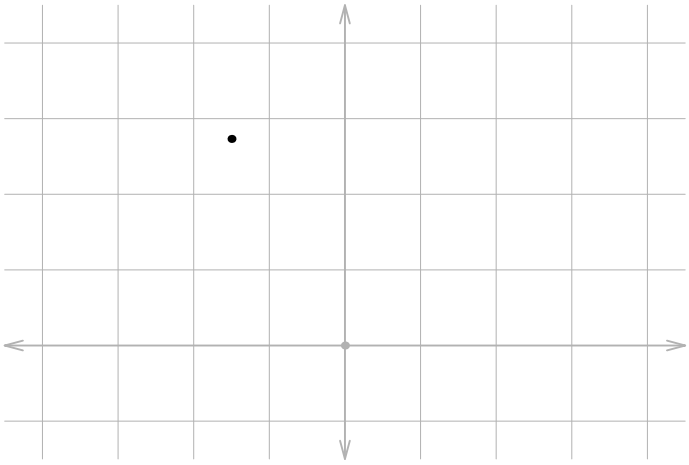
Príklad 5:

Nakreslite bod $(-\frac{3}{2}, 1+\sqrt{3})$.

Riešenie

Pri práci s číslami môžeme používať aj bežné matematické operácie, vzniknuté výrazy rešpektujú všeobecne dohodnuté pravidlá ich priorít. Tu sú ako ukážky použité pri prvej súradnici delenie (zapísané očakávane pomocou `/`) a opačné číslo (`-`) a pri druhej odmocnina (programátorsky `sqrt`) a súčet (`+`). Program teda vyzerá takto:

```
kresliSiet(-4,4, -1,4);

dot((-3/2,1+sqrt(3)));
```



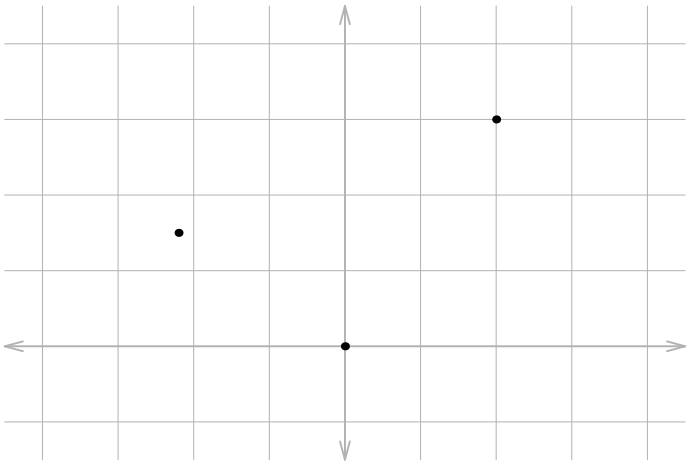
Príklad 6:

Nakreslite body $(0,0)$, $(2,3)$ a $(-2{,}2, 1{,}5)$.

Riešenie

Náš program teraz bude mať tri veľmi podobné príkazy, každý z nich bude zodpovedný za vykreslenie jedného bodu. Kvôli prehľadnosti výsledného kódu napíšeme každý z nich do osobitného riadku:

```
kresliSiet(-4,4, -1,4);

dot((0,0));
dot((2,3));
dot((-2.2,1.5));
```



Úlohu sme síce splnili, ale trochu strácame prehľad, ktorý bod zodpovedá ktorému riadku. Uvedomme si, že body obvykle znázorňujeme preto, lebo ich potrebujeme na ďalšie konštrukcie. Aby sme sa na ne mohli pri týchto konštrukciách odvolávať, potrebujeme ich na obrázku označiť nejakým textom. Treba pritom dbať na to, aby označenia nezakrývali dôležité časti obrázka a aby boli rozmiestnené esteticky.

Príklad 7:

Zvoľte si bod a označte ho zvoleným označením.

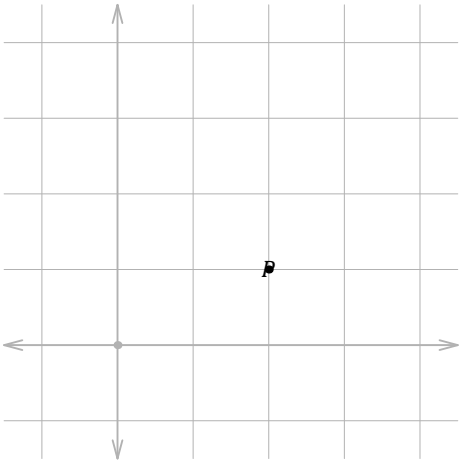
Riešenie

Na vypísanie textu slúži príkaz `label` (značka). Jeho prvým vstupom je vypisovaný text (ak ide o konštantu, tak musí byť v úvodzovkách (`"`)) a druhým miesto (čiže bod), kde sa má tento text vypísať.

Povedzme, že náš bod bude $(2, 1)$ a bude sa volať P.

Všimnime si, že toto označenie je šikmým písmom, a to podľa pravidiel matematickej sadzby. Aby sme túto normu dodržali, stačí využiť schopnosti typografického systému TEX, ktorý s programom Asymptote úzko spolupracuje, a to tak, že označenie obalíme z oboch strán znakom `$`:

```
kresliSiet(-1,4, -1,4);

dot((2,1));
label("$P$",(2,1));
```



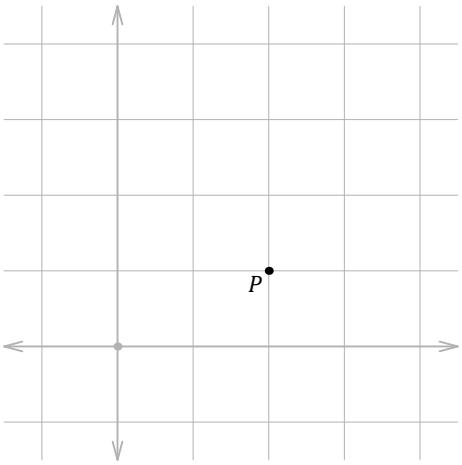
Žiaľ, nedopadlo to úplne najlepšie, lebo označenie zakrýva označovaný bod. Potrebujeme ho trocha premiestniť.

Ak chceme označenie z určeného miesta mierne posunúť, do príkazu `label` pridáme ďalší vstup. Jeho základné hodnoty sú anglické značky základných i vedľajších svetových strán s prirodzenou interpretáciou:

- `N` –„north“ (sever),
- `NE` –„north-east“ (severovýchod),
- `E` –„east“ (východ),
- `SE` –„south-east“ (východ),
- `S` –„south“ (juh),
- `SW` –„south-west“ (juhozápad),
- `W` –„west“ (západ),
- `NW` –„north-west“ (severozápad).

Ak chceme označenie nášho bodu zobraziť od neho na juhozápad (čiže vľavo dolu), náš program bude vyzerať takto:

```
kresliSiet(-1,4, -1,4);

dot((2,1));
label("$P$",(2,1),SW);
```



Príkazy `dot` a `label` týkajúce sa toho istého bodu môžeme združiť do jedného tak, že vstupy druhého presunieme do prvého a druhý zrušíme.

Konečná verzia nášho programu teda bude vyzerať takto:

```
kresliSiet(-1,4, -1,4);

dot("$P$",(2,1),SW);
```

Úloha 1:

Nakreslite vrcholy A, B, C, D jednotkového štvorca $ABCD$ umiestneného do rohu prvého kvadrantu. (Pamätajte na estetické rozmiestnenie ich označení.)

1.2 Úsečky

Príklad 8:

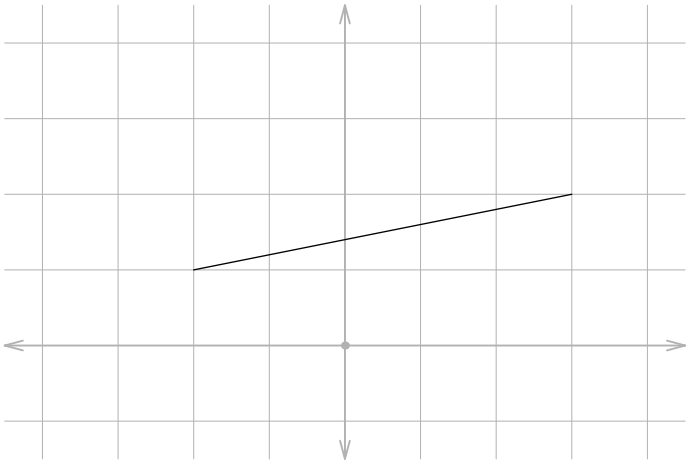
Nakreslite úsečku s krajnými bodmi $(-2, 1)$ a $(3, 2)$.

Riešenie

- Úsečka s krajnými bodmi A a B sa zapisuje v tvare `A--B`.
- Na nakreslenie krivky (napríklad úsečky) sa používa príkaz `draw` (kresli), ktorý je nasledovaný krivkou v okrúhlych zátvorkách.

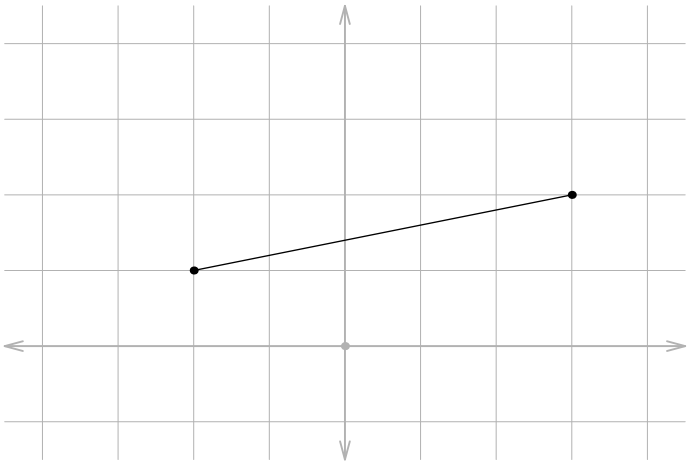
V našom prípade ide o úsečku `(-2,1)--(3,2)`, nakreslíme ju teda takto:

```
kresliSiet(-4,4, -1,4);

draw((-2,1)--(3,2));
```



Úsečka bez zvýraznenia krajných bodov však pôsobí akosi bezprizorne. Pri jej kreslení preto budeme vždy explicitne znázorňovať aj jej krajné body.

```
kresliSiet(-4,4, -1,4);

draw((-2,1)--(3,2));

dot((-2,1));
dot((3,2));
```



Príklad 9:

Nakreslite úsečku s krajnými bodmi

a) $(0, 0)$ a $(3, 2)$;

b) $(2, -1)$ a $(3, 2)$;

c) $(3, 1)$ a $(3, 2)$.

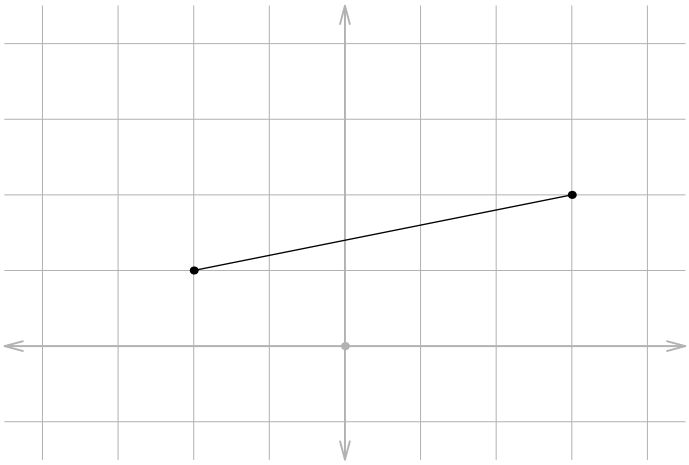
Riešenie

Pri riešení takmer každej úlohy obvykle šetríme svoj čas i energiu tým, že program nepíšeme stále znova od začiatku, ale minimalisticky upravujeme kód nejakej podobnej úlohy. Táto úloha sa líši od predchádzajúceho príkladu len zmenou prvého bodu. Preto v jeho riešení

```
kresliSiet(-4,4, -1,4);

draw((-2,1)--(3,2));

dot((-2,1));
dot((3,2));
```

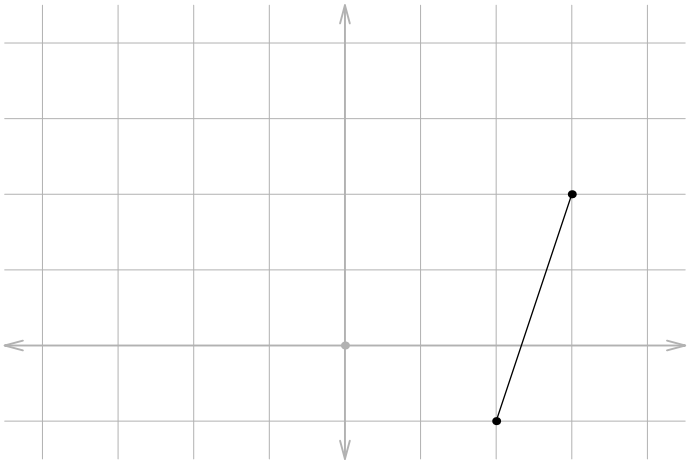
stačí najprv hodotu `(-2,1)` prepísať na `(0,0)` (bez potreby meniť čokoľvek ostatné):

```
kresliSiet(-4,4, -1,4);

draw((0,0)--(3,2));

dot((0,0));
dot((3,2));
```



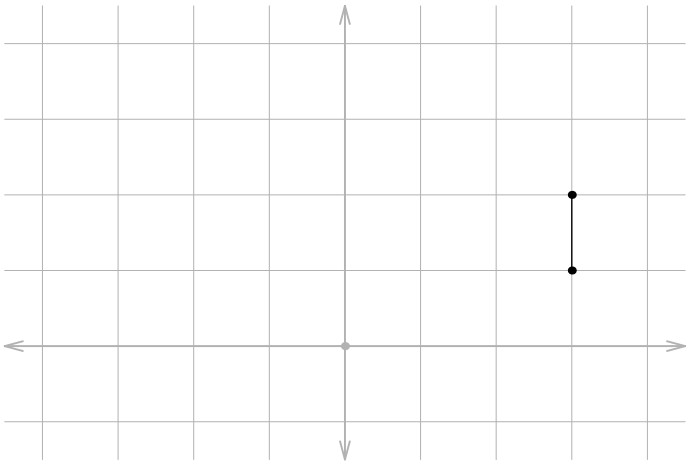
potom na `(2,-1)`:

```
kresliSiet(-4,4, -1,4);

draw((2,-1)--(3,2));

dot((2,-1));
dot((3,2));
```



a napokon na `(3,1)`:

```
kresliSiet(-4,4, -1,4);

draw((3,1)--(3,2));

dot((3,1));
dot((3,2));
```



Vzniká tu však prirodzená otázka, prečo musíme robiť zmenu jedného bodu zakaždým až na dvoch rôznych miestach. Nepríjemnosť tejto duplicity sa (zámerným) opakovaným riešením úloh toho istého typu čoraz viac zvýrazňuje. Nechuť upravovať ten istý údaj na viacerých miestach netreba hneď vnímať negatívne ako nejakú lenivosť. Zaslúži si skôr pochvalu, lebo nás upozorňuje na jeden zo *základných princípov práce s dátami*:

Každý údaj má byť zapísaný na jedinom mieste.

Máloktorý údaj je totiž taký stabilný, že ho netreba nikdy aktualizovať. V prípade nutnosti zmeny treba vyhľadať všetky jeho výskyty a patrične ich upraviť. Problém však je, že týchto výskytov nezriedka býva nepreberné množstvo a časom môžeme stratiť prehľad, kde všade treba aktualizáciu urobiť. Navyše sa môže stať, že omylom modifikujeme aj údaj, ktorý má zhodou okolností rovnakú hodnotu ako náš. Vzniká tak veľké nebezpečenstvo *nekonzistencie dát*.

V našom prípade toto nebezpečenstvo nie je veľké – stačí dávať pozor na to, aby sa prvý bod z prvého riadku zhodoval s bodom z druhého riadku. Prečo si však pridávať zbytočné starosti, keď takouto kontrolou možno poveriť program?

Tento problém sa v programovacích jazykoch (včítane Asymptote) rieši zavedením *premenných*. Každú premennú si môžeme predstaviť ako akúsi skrinku, do ktorej uložíme nejakú hodnotu. Ak potom chceme pracovať s touto hodnotou, stačí osloviť premennú a tá nám svoju hodnotu hneď prezradí.

Každý ukladaný údaj má pritom istú štruktúru, ktorá si vyžaduje skrinku istého tvaru a objemu. Preto má (v mnohých programovacích jazykoch včítane Asymptote) každá premenná určený svoj *dátový typ*, ktorý ju predurčuje na ukladanie údajov v zodpovedajúcej štruktúre.

Pod *deklaráciou* premennej rozumieme jej pomenovanie a stanovenie jej dátového typu, ale i vyhlásenie (a teda aj prísľub), že ju ďalej budeme používať. Uvedomme si ďalší princíp práce s údajmi:

Zavedenie pomenovania, ktoré sa potom nepoužije, je nezmyselné.

Mimochodom, všimnime si, ako často (a dokonca úmyselne) sa toto pravidlo porušuje v učebniciach matematiky…

- Ak chceme deklarovať premennú s menom p a dátovým typom t, napíšeme `t p`.
- Ak chceme premennú s menom p naplniť hodnotou h zodpovedajúceho dátového typu, napíšeme príkaz `p = h`.
- Príkaz deklarácie premennej p dátového typu t a jej naplnenie hodnotou h sa často spája do združeného príkazu `t p = h`.

V našom prípade potrebujeme opakovane pracovať s krajným bodom úsečky, ktorý je, ako vieme, reprezentovaný dvojicou čísel.

Na uloženie dvojice reálnych čísel sa v Asymptote používa dátový typ `pair` (dvojica).

Máme dva body, zavedieme teda dve premenné takéhoto typu, povedzme `A` a `B`. Do prvej uložíme bod `(0,0)` a každý výskyt bodu `(0,0)` nahradíme `A`. Analogicky to urobíme s druhou premennou. Namiesto kódu

```
kresliSiet(-4,4, -1,4);

draw((0,0)--(3,2));

dot((0,0));
dot((3,2));
```

tak dostávame

```
kresliSiet(-4,4, -1,4);

pair A = (0,0);
pair B = (3,2);

draw(A--B);

dot(A);
dot(B);
```

Je síce pravda, že kód sa trochu predĺžil, ale táto investícia sa rýchlo vráti v rýchlosti jeho modifikácie na kód podobnej úlohy, stačí zmeniť vyznačenú časť

```
kresliSiet(-4,4, -1,4);

pair A = (0,0);
pair B = (3,2);

draw(A--B);

dot(A);
dot(B);
```

takto:

```
kresliSiet(-4,4, -1,4);

pair A = (2,-1);
pair B = (3,2);

draw(A--B);

dot(A);
dot(B);
```

resp. takto:

```
kresliSiet(-4,4, -1,4);

pair A = (3,1);
pair B = (3,2);

draw(A--B);

dot(A);
dot(B);
```

Príklad 10:

Nakreslite úsečku s krajnými bodmi $(-2, 1)$ a $(3, 2)$

a) čiarkovane;
b) bodkovane;
c) červenou;
d) tučne;
e) červenou a čiarkovane;
f) tučne, červenou a čiarkovane;
g) s orientáciou od prvého k druhému bodu;
h) červenou a s orientáciou od prvého k druhému bodu.

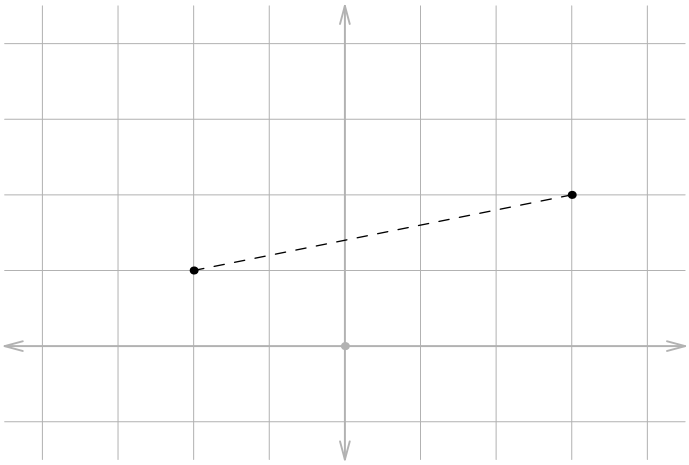
Riešenie

- Namiesto preddefinovanej plnej čiary môžeme nakresliť čiarkovanú alebo bodkovanú, a to tak, že ako jeden zo vstupov príkazu `draw` dáme slovo `dashed` (čiarkovaný), resp. `dotted` (bodkovaný).
- Obvyklá čierna farba čiary (ale aj písma) sa dá zmeniť pridaním vstupu s anglickým menom novej farby.
- Hrúbku čiary môžeme ovplyvniť pridaním vstupu `linewidth` (šírka čiary) doplnenom veľkosťou tejto hrúbky.
- Tieto zmeny môžeme aj kombinovať, a to tak, že medzi ne vložíme znak `+`.
- Ak chceme, aby sa krivka končila šípkou, príkaz `draw` bude mať ďalší vstup, a to `Arrow` (šípka).

Naše úlohy teda môžeme vyriešiť postupne takto:

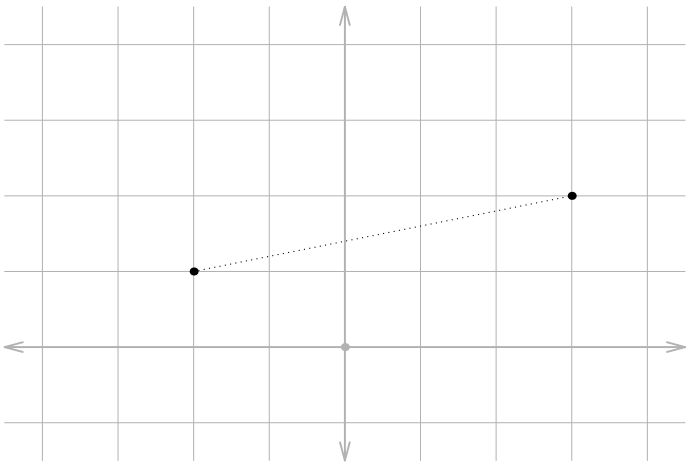
a)

```
kresliSiet(-4,4, -1,4);

pair A = (-2,1);
pair B = (3,2);

draw(A--B,dashed);

dot(A);
dot(B);
```



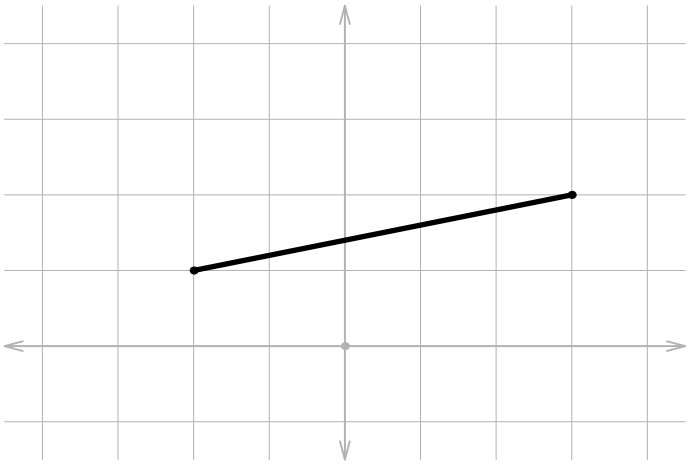
b)

```
kresliSiet(-4,4, -1,4);

pair A = (-2,1);
pair B = (3,2);

draw(A--B,dotted);

dot(A);
dot(B);
```



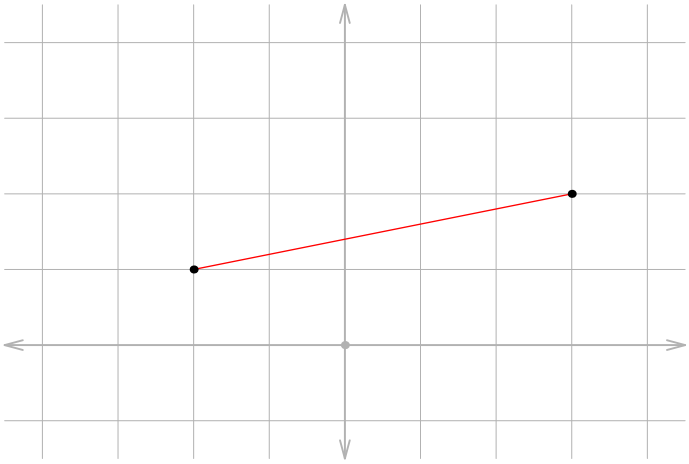
c)

```
kresliSiet(-4,4, -1,4);

pair A = (-2,1);
pair B = (3,2);

draw(A--B,linewidth(2));

dot(A);
dot(B);
```



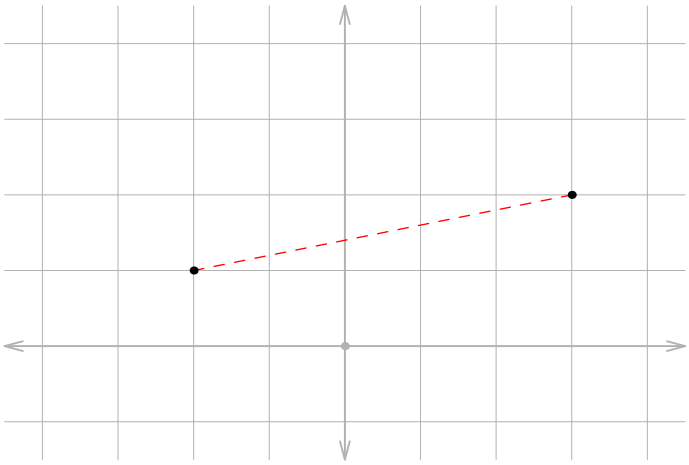
d)

```
kresliSiet(-4,4, -1,4);

pair A = (-2,1);
pair B = (3,2);

draw(A--B,red);

dot(A);
dot(B);
```



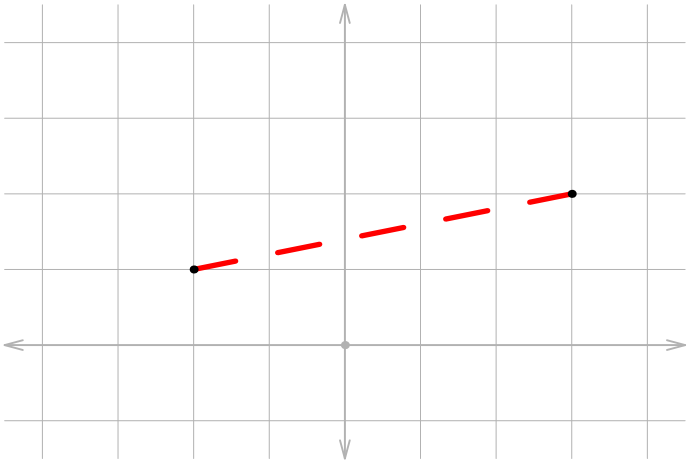
e)

```
kresliSiet(-4,4, -1,4);

pair A = (-2,1);
pair B = (3,2);

draw(A--B,red+dashed);

dot(A);
dot(B);
```



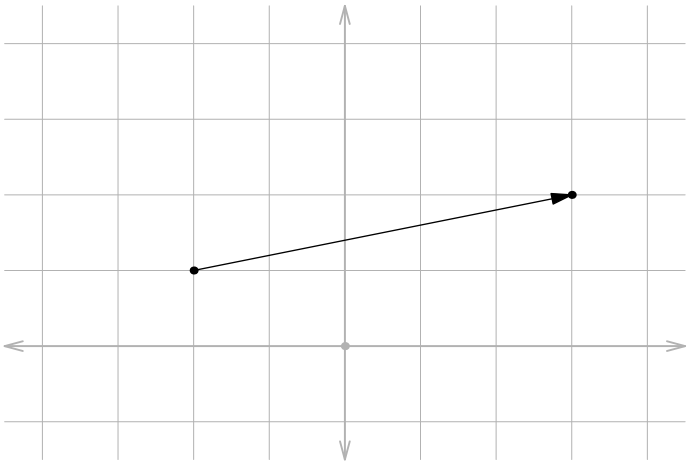
f)

```
kresliSiet(-4,4, -1,4);

pair A = (-2,1);
pair B = (3,2);

draw(A--B,red+dashed+linewidth(2));

dot(A);
dot(B);
```



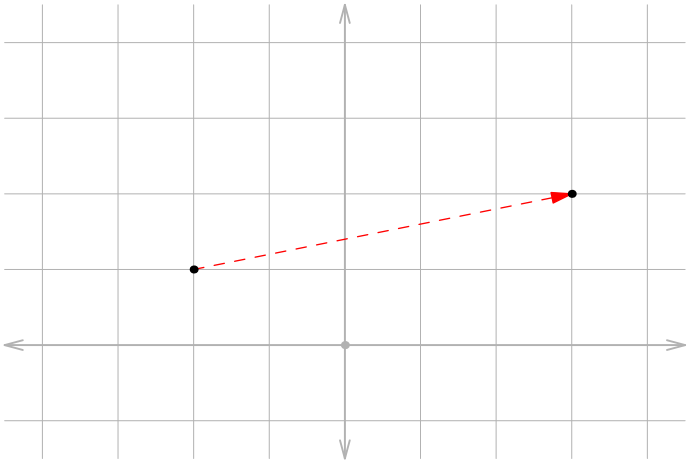
g)

```
kresliSiet(-4,4, -1,4);

pair A = (-2,1);
pair B = (3,2);

draw(A--B,Arrow);

dot(A);
dot(B);
```



h)

```
kresliSiet(-4,4, -1,4);

pair A = (-2,1);
pair B = (3,2);

draw(A--B,red,Arrow);

dot(A);
dot(B);
```



Výskumná úloha 2:

Prečo najprv kreslíme úsečku a až potom jej krajné body?

1.3 Lomené krivky

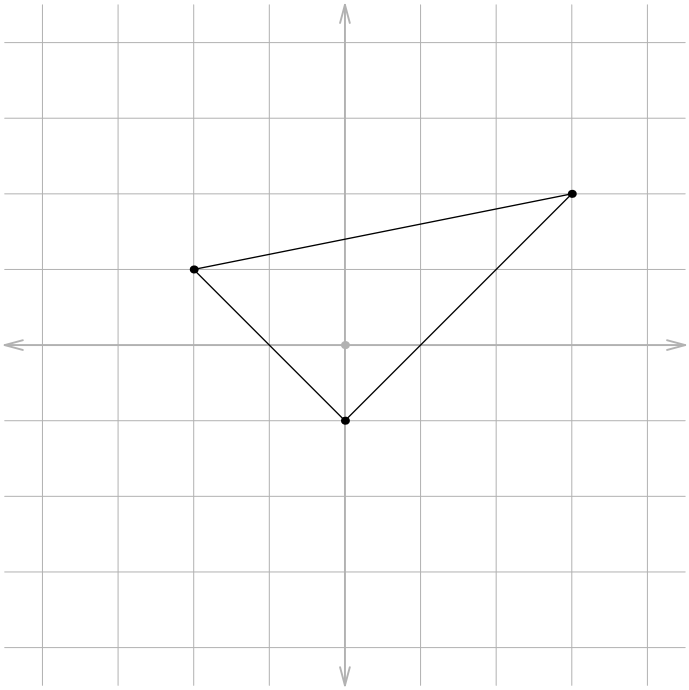
Príklad 11:

Vyfarbite červenou trojuholník s vrcholmi $(0,-1)$, $(-2,1)$ a $(3,2)$.

Riešenie

Ak by úloha žiadala nakresliť len obvod tohto trojuholníka, stačilo by si uvedomiť, že ten je tvorený troma úsečkami. Vrcholy by sme označili postupne `A`, `B`, `C`, potom nakreslili ich spojnice a napokon aj samotné body takto:

```
kresliSiet();

pair A = (0,-1);
pair B = (-2,1);
pair C = (3,2);

draw(A--B);
draw(B--C);
draw(C--A);

dot(A);
dot(B);
dot(C);
```



My však potrebujeme pracovať s obvodom ako celkom. Keďže strany trojuholníka majú spoločné krajné body, jeho obvod je uzavretá lomená krivka.

- Ak P_0, P_1, ..., P_n sú body, lomenú krivku $P_0 \ldots P_n$ zapíšeme v tvare `P0--P1--...--Pn`.
- Ak ide o uzavretú lomenú krivku, čiže ak $P_n = P_0$, píšeme `P0--P1--...--Pn-1--cycle`.

V našom prípade teda ide o uzavretú krivku `A--B--C--cycle`. Program preto prepíšeme takto:

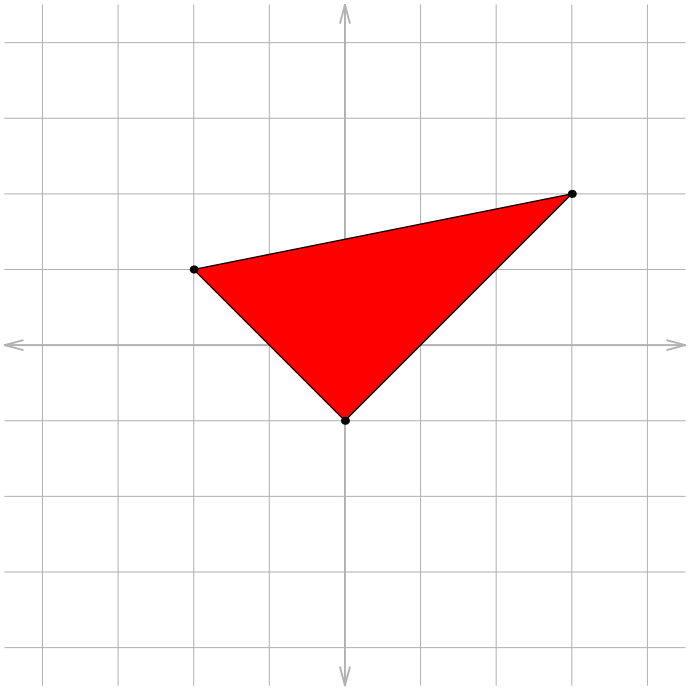
```
kresliSiet();

pair A = (0,-1);
pair B = (-2,1);
pair C = (3,2);

draw(A--B--C--cycle);

dot(A);
dot(B);
dot(C);
```

Krivku však treba aj vyfarbiť.

Na vyplnenie uzavretej krivky slúži príkaz `fill` (vyplň), ktorého prvý vstup je samotná krivka a druhý vypĺňajúca farba.

Môžeme teda písať:

```
kresliSiet();

pair A = (0,-1);
pair B = (-2,1);
pair C = (3,2);

fill(A--B--C--cycle,red);
draw(A--B--C--cycle);

dot(A);
dot(B);
dot(C);
```



Ako vidíme, príkazy `fill` a `draw` majú za vstup tú istú krivku, je tu teda istá duplicita.

Ak chceme pracovať s krivkou ako celkom, je vhodné uložiť ju do premennej typu `path` (cesta).

Spomínanú duplicitu teda môžeme odstrániť takto:

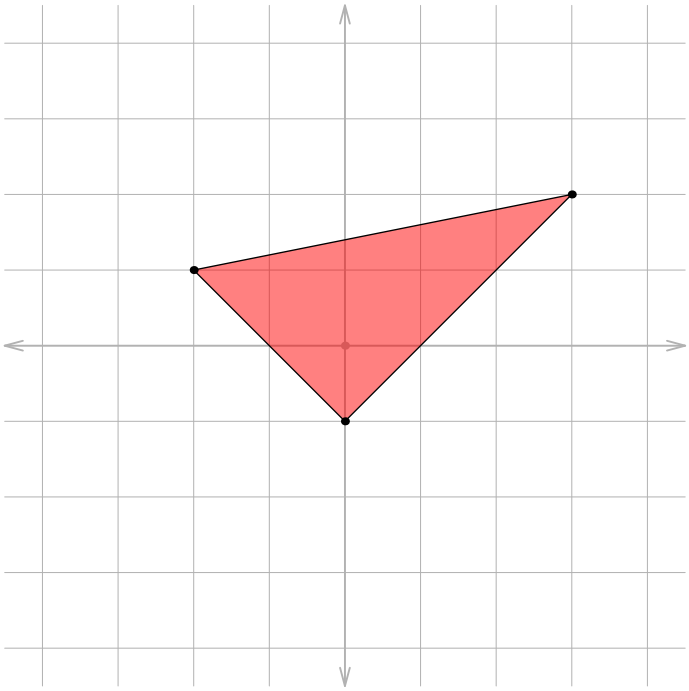
```
kresliSiet();

pair A = (0,-1);
pair B = (-2,1);
pair C = (3,2);

path t = A--B--C--cycle;

fill(t,red);
draw(t);

dot(A);
dot(B);
dot(C);
```

Ešte malý technický problém: trojuholník je vyfarbený takou sýtou farbou, že nevidieť body za ním. Bude preto dobre ho trochu spriehľadniť.

Stupeň (ne)priehľadnosti môžeme stanoviť pomocou príkazu `opacity` (nepriehľadnosť), ktorého vstup je číslo medzi `0` a `1` – čím menšie číslo, tým väčšia priehľadnosť.

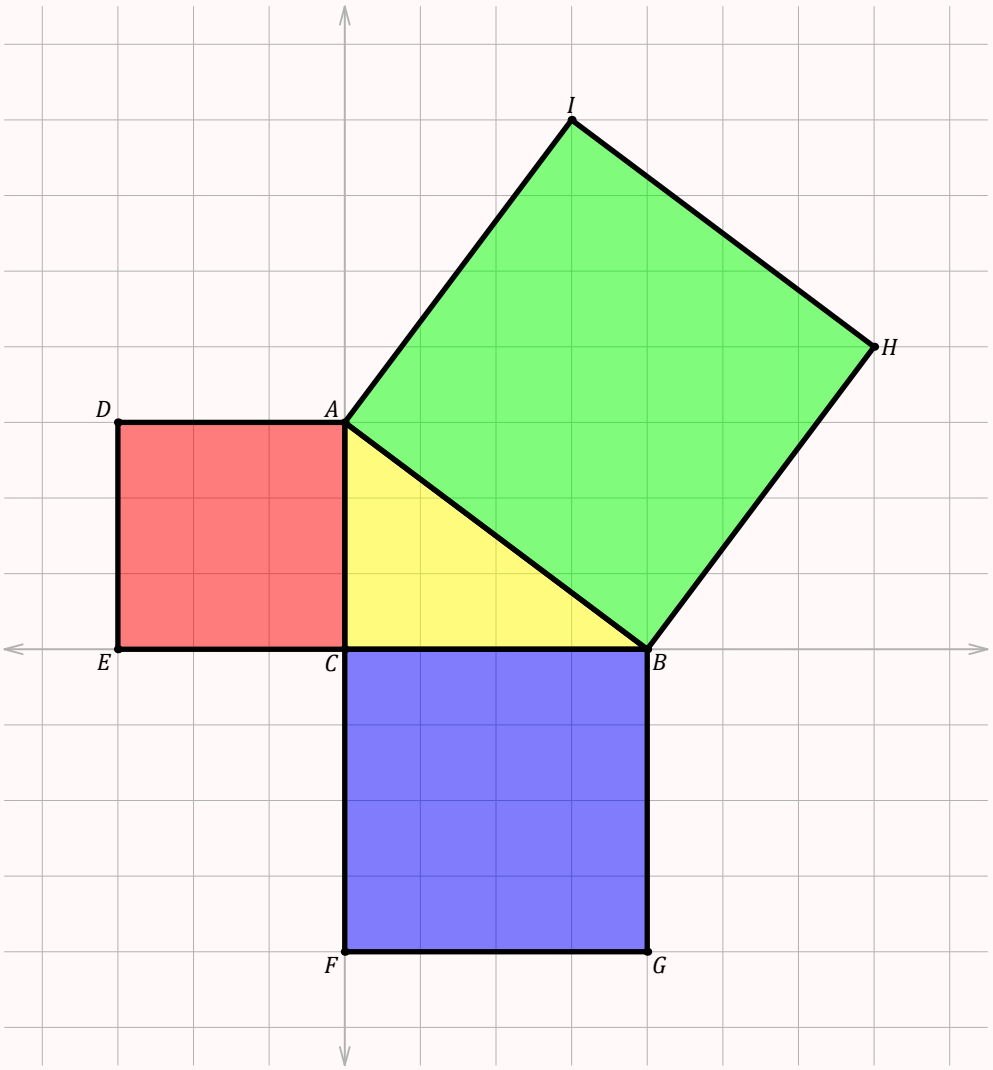
Zvolíme priemernú hodnotu `.5`:

```
kresliSiet();

pair A = (0,-1);
pair B = (-2,1);
pair C = (3,2);

path t = A--B--C--cycle;

fill(t,red+opacity(.5));
draw(t);

dot(A);
dot(B);
dot(C);
```



Úloha 2:

Nakreslite nasledujúci obrázok:



V predchádzajúcej úlohe je prirodzené pre každý z vrcholov zaviesť vlastnú rovnomennú premennú. Treba si však uvedomiť, že premenná `E` (tiež je typu `pair`) je už vyhradená na iné účely (umiestňovanie označení). Pre bod E tak treba použiť inú premennú (napríklad `EE`).

2

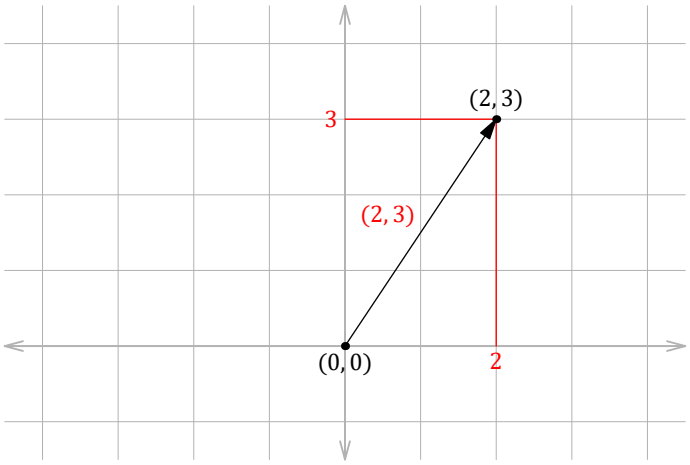
Zobrazenia

2.1 Posunutie

Vektor v rovine je plne charakterizovaný koncovým bodom svojej *základnej polohy* (t. j. umiestnenia, v ktorom jeho začiatočný bod splýva s počiatkom súradnicovej sústavy). Pod jeho *súradnicami* rozumieme práve súradnice tohto koncového bodu. Hovoríme tak o x*-ovej* a y-ovej súradnici vektora. Vektor, ktorý má v základnej polohe koncový bod (a, b), budeme preto označovať rovnako (a, b).

V stredoškolskej matematike sa označenia bodu a vektora obvykle rozlišujú použitými zátvorkami, takže naše jednotné označovanie okrúhlymi zátvorkami sa môže zdať nekorektné. Uvedomme si však, že v skutočnosti pracujeme aj v jednom a v druhom prípade s dvojicou reálnych čísel a to, či túto usporiadanú dvojicu chápeme ako bod, alebo ako vektor (alebo dokonca ako komplexné číslo), je len našou interpretáciou. Je vecou našej šikovnosti zvoliť si také chápanie usporiadanej dvojice, ktoré zodpovedá tomu, čo chceme vyjadriť. Jednotné označovanie navyše poskytuje jednu veľkú psychologickú výhodu: umožňuje napríklad sčítavať bod s vektorom bez zlého pocitu, že miešame jablká s hruškami.

Vektor $(2, 3)$ na obrázku má x-ovú súradnicu 2 a y-ovú súradnicu 3:

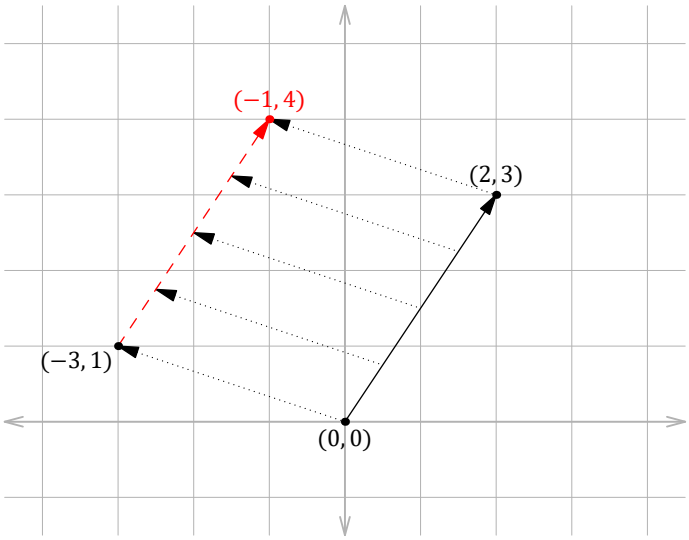


Príklad 12:

Kam sa zobrazí bod $(-3, 1)$ v posunutí o vektor $(2, 3)$?

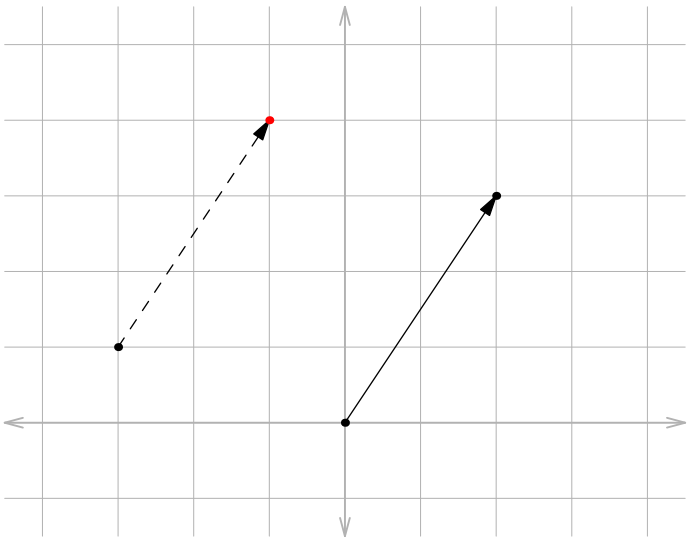
Riešenie

Vektor $(2, 3)$ premiestnime (bez zmeny orientácie) tak, aby jeho začiatok splynul s bodom $(-3, 1)$. Obrazom tohto bodu v posunutí o tento vektor bude potom koniec takto premiestneného vektora čiže (ako zistíme z obrázka) bod $(-1, 4)$.



Posúvaný bod $(-3, 1)$ vložíme do premennej `A`, vektor $(2, 3)$ do premennej `V` a výsledný bod $(-1, 4)$ do premennej `AA`. Budeme potrebovať aj počiatok súradnicovej sústavy $(0, 0)$, ktorý vložíme do premennej `O`. Všetky tieto premenné (bez ohľadu na to, či ide o bod, alebo o vektor) budú typu `pair`. Ďalšími príkazmi nakreslíme vektor v základnej polohe a jeho premiestnenie do bodu, ktorý bude vzorom výsledku.

```
kresliSiet(-4,4, -1,5);

pair O = (0,0);
pair A = (-3,1);
pair V = (2,3);
pair AA = (-1,4);

draw(O--V,Arrow);
draw(A--AA,dashed,Arrow);

dot(O);
dot(A);
dot(V);
dot(AA,red);
```



Príklad 13:

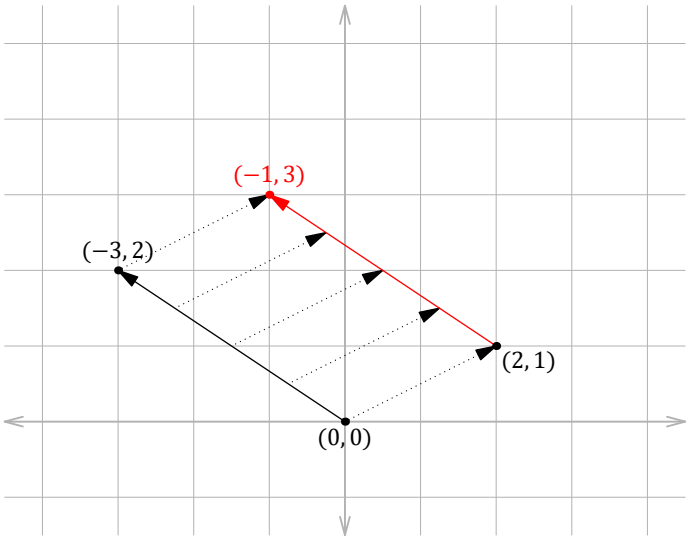
Kam sa zobrazí bod $(2, 1)$ v posunutí o vektor $(-3, 2)$?

Riešenie

Opäť tu máme nápadnú podobu s tým, čo sme už raz riešili. Stačí vziať riešenie predchádzajúcej úlohy

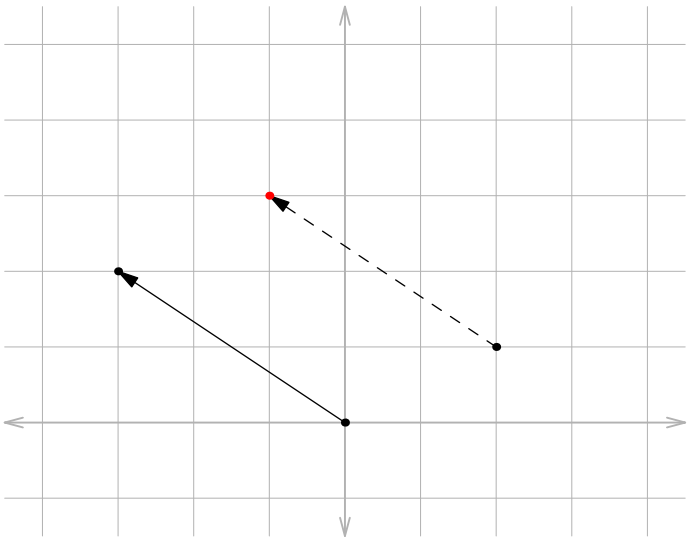
```
kresliSiet(-4,4, -1,5);

pair O = (0,0);
pair A = (-3,1);
pair V = (2,3);
pair AA = (-1,4);

draw(O--V,Arrow);
draw(A--AA,red+dashed,Arrow);

dot(O);
dot(A);
dot(V);
dot(AA,red);
```

a upraviť v ňom príslušné číselné hodnoty: Premennú `A`, označujúcu posúvaný bod, nastavíme na `(2,1)` a premennú vektora `V` zmeníme na `(-3,2)`. Nová hodnota premennej `AA` však v zadaní explicitne nie je, treba ju nejako zistiť. Podobným obrázkom zistíme, že je to $(-1, 3)$:



Dostávame tak takýto program:

```
kresliSiet(-4,4, -1,5);

pair O = (0,0);
pair A = (2,1);
pair V = (-3,2);
pair AA = (-1,3);

draw(O--V,Arrow);
draw(A--AA,dashed,Arrow);

dot(O);
dot(A);
dot(V);
dot(AA,red);
```



V predchádzajúcich dvoch príkladoch však cítiť istú disproporciu: Kým samotný problém má dva vstupy (vzor a vektor), program, ktorý ho rieši, má vstupy až tri (vzor, vektor a obraz). Je zatiaľ našou zodpovednosťou, aby bola ručne zapísaná hodnota tretieho vstupu konzistentná s prvými dvoma.

Nemôžeme túto nepríjemnú povinnosť prenechať programu? Samozrejme, že môžeme, musíme však nájsť postup, ako túto tretiu hodnotu získať z prvých dvoch, a zapísať ho v programovacom jazyku.

Výskumná úloha 3:

Nájdite vzťah medzi súradnicami vektora, vzoru a jeho obrazu v posunutí o tento vektor.

Všimnime si, že v oboch predchádzajúcich príkladoch platí:

- Prvá súradnica obrazu je súčtom prvej súradnice vzoru a prvej súradnice vektora:
  - Príklad 12: $-1 = (-3) + 2$.
  - Príklad 13: $-1 = 2 + (-3)$.
- Druhá súradnica obrazu je súčtom druhej súradnice vzoru a druhej súradnice vektora:
  - Príklad 12: $4 = 1 + 3$.
  - Príklad 13: $3 = 1 + 2$.

Aby sme nemuseli pracovať s oboma súradnicami osobitne, môžeme definovať súčet (a predvídavo i rozdiel) dvojíc reálnych čísel:

$$(a, b) \pm (c, d) = (a \pm c, b \pm d).$$

Sčítaním vzoru a vektora v našom príklade tak dostávame:

- Príklad 12: $(-1,4) = (-3,1) + (2,3)$.
- Príklad 13: $(-1,3) = (-3,1) + (2,3)$.

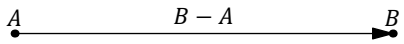
Zovšeobecnením predchádzajúcich pozorovaní dostávame, že obrazom bodu (x,y) v posunutí o vektor (u,v) je bod $(x+u, y+v)$. Alebo inak: Ak A je bod a $\vec{v}$ je vektor, ich súčet $A+\vec{v}$ možno interpretovať ako bod, ktorý je obrazom bodu A v posunutí o vektor $\vec{v}$.

Obrazom bodu A v posunutí o vektor $\overrightarrow{AB}$ je, samozrejme, bod B. Podľa predchádzajúceho odseku tak môžeme písať

$$\boxed{A + \overrightarrow{AB} = B}$$

alebo krajšie

$$\boxed{\overrightarrow{AB} = B - A}$$

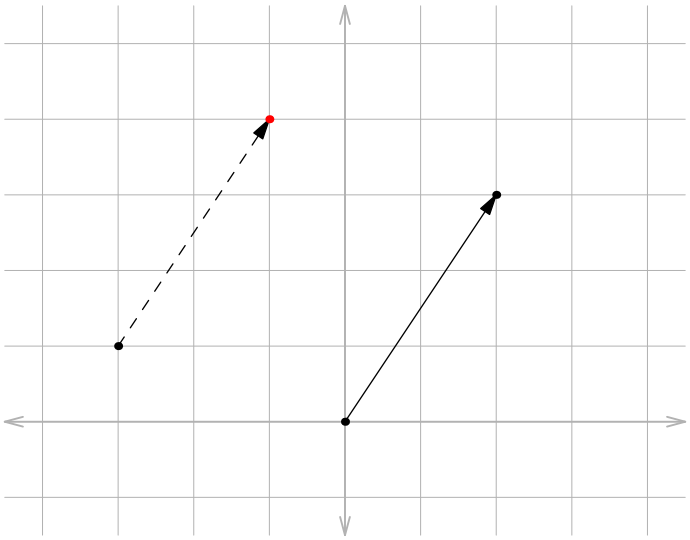


S týmito poznatkami sa vráťme k predošlým dvom príkladom: Namiesto pôvodného kódu

```
kresliSiet(-4,4, -1,5);

pair O = (0,0);
pair A = (-3,1);
pair V = (2,3);
pair AA = (-1,3);

draw(O--V,Arrow);
draw(A--AA,dashed,Arrow);

dot(O);
dot(A);
dot(V);
dot(AA);
```

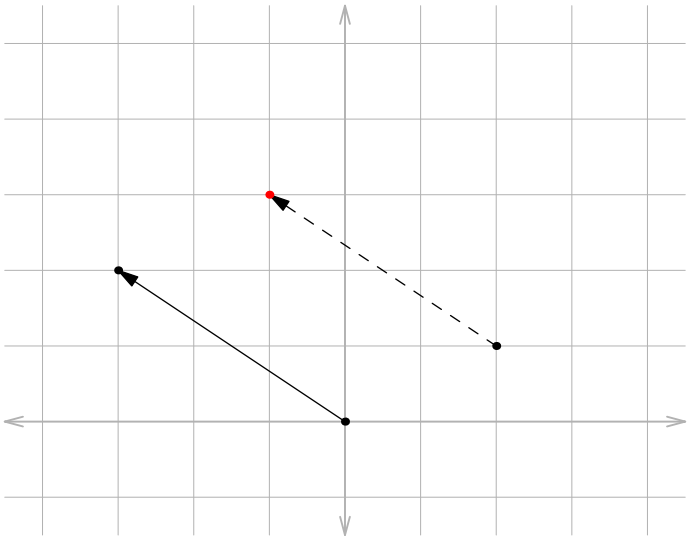
tak v príklade 12 budeme mať

```
kresliSiet(-4,4, -1,5);

pair O = (0,0);
pair A = (-3,1);
pair V = (2,3);
pair AA = A+V;

draw(O--V,Arrow);
draw(A--AA,dashed,Arrow);

dot(O);
dot(A);
dot(V);
dot(AA,red);
```



Ak chceme vyriešiť príklad 13, stačí modifikovať iba premenné A a V, hodnota premennej AA sa upraví automaticky:

```
kresliSiet(-4,4, -1,5);

pair O = (0,0);
pair A = (2,1);
pair V = (-3,2);
pair AA = A+V;

draw(O--V,Arrow);
draw(A--AA,dashed,Arrow);

dot(O);
dot(A);
dot(V);
dot(AA,red);
```



Tým sme odstránili nesúlad v počte „stupňov voľnosti“ riešeného problému a jemu zodpovedajúceho programu.

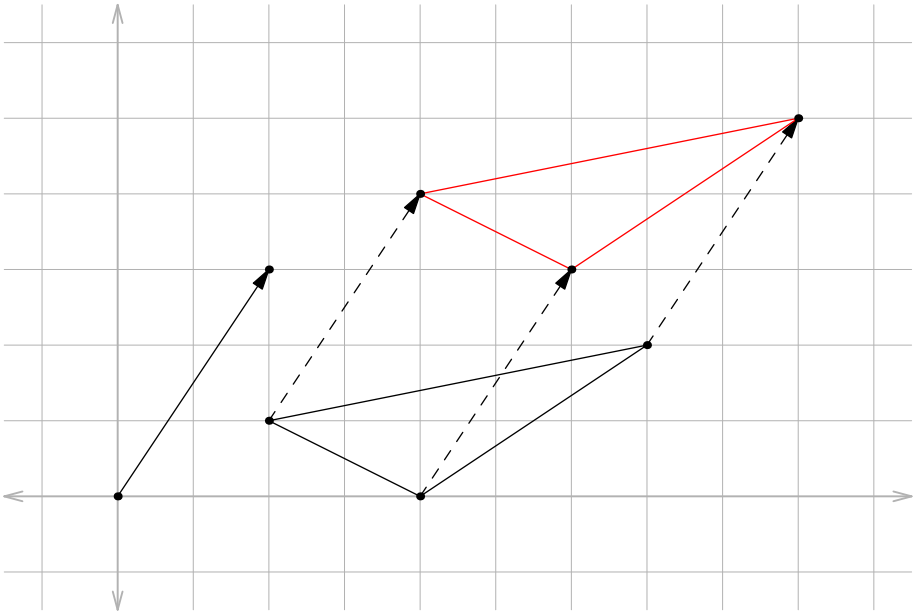
Príklad 14:

Kam sa zobrazí trojuholník s vrcholmi $(2,1)$, $(4,0)$ a $(7,2)$ v posunutí o vektor $(2,3)$?

Riešenie

Vektor posunutia označme V, vrcholy pôvodného trojuholníka označme postupne A, B, C a ich obrazy v posunutí o vektor V postupne AA, BB, CC:

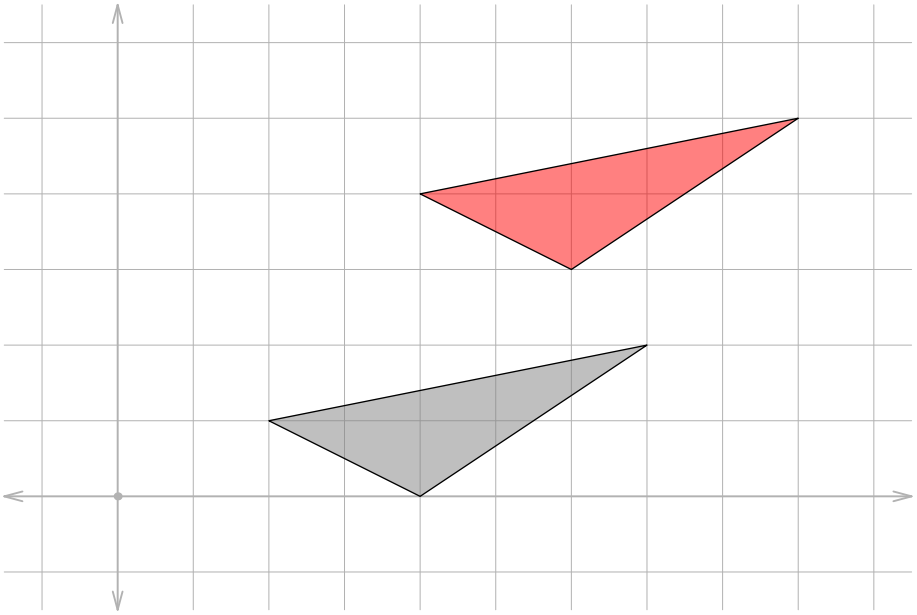
```
kresliSiet(-1,10, -1,6);

pair O = (0,0);
pair V = (2,3);

pair A = (2,1);
pair B = (4,0);
pair C = (7,2);

pair AA = A+V;
pair BB = B+V;
pair CC = C+V;

draw(O--V,Arrow);

draw(A--AA,dashed,Arrow);
draw(B--BB,dashed,Arrow);
draw(C--CC,dashed,Arrow);

draw(A--B--C--cycle);
draw(AA--BB--CC--cycle,red);

dot(O);

dot(V);

dot(A);
dot(B);
dot(C);

dot(AA);
dot(BB);
dot(CC);
```



Ak nie je nutné pracovať s každým presúvaným bodom osobitne, Asymptote ponúka alternatívu – príkaz `shift` (posuň), ktorý celý útvar posunie naraz. Pod zápisom `shift(`$\vec{v}$`)*`M rozumieme posunutie bodu alebo množiny bodov M o vektor $\vec{v}$.

Riešenie našej úlohy teda môže vyzerať aj takto:

```
kresliSiet(-1,10, -1,6);

pair V = (2,3);

pair A = (2,1);
pair B = (4,0);
pair C = (7,2);

path t = A--B--C--cycle;
path tt = shift(V)*t;

fill(t,gray+opacity(.5));
draw(t);

fill(tt,red+opacity(.5));
draw(tt);
```



Predchádzajúci príklad ukazuje, že na obrázku sa nemusia ocitnúť všetky prvky konštrukcie. Príkladom je koncový bod vektora (napokon, i vektor samotný). Nastáva tu teda dôležité mentálne oddelenie konštrukcie od jej vykreslenia, ktoré pri klasickej konštrukcii na papieri nepozorujeme.

Úloha 3:

Posuňte jednotkový štvorec o vektor $(-3, 4)$.

Príklad 15:

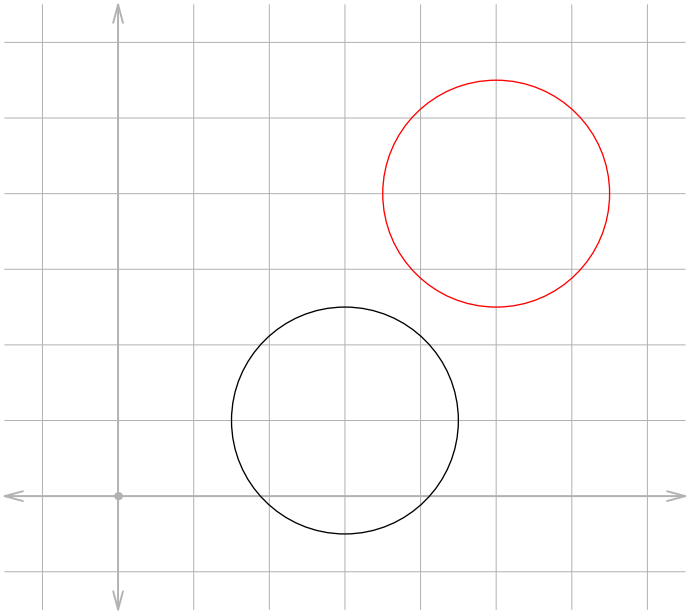
Posuňte kružnicu so stredom $(3, 1)$ s polomerom 1,5 o vektor $(2, 3)$.

Riešenie

Kružnica so stredom S a polomerom r je objekt typu `path`, možno ju získať príkazom `circle` (kružnica) s parametrami S a r, t. j. `circle(`S`,`r`)`, pričom S je typu `pair` a r je (nezáporné) reálne číslo čiže typu `real` (reálne číslo).

V programe definujeme najprv pôvodnú kružnicu ako krivku `k` so stredom `SS` a polomerom `r` a potom jej obraz ako krivku `kk`, ktorá vznikne posunutím krivky `k` o vektor `V`.

```
kresliSiet(-1,7, -1,6);

pair V = (2,3);

pair SS = (3,1);
real r = 1.5;

path k = circle(SS,r);
path kk = shift(V)*k;

draw(k);
draw(kk,red);
```



Úloha 4:

Posuňte kruh so stredom $(3, 1)$ s polomerom 1,5 o vektor $(2, 3)$.

Výskumná úloha 4:

Nech a je útvar a $\vec{v}$ je vektor.

Označme b obraz útvaru a v posunutí o $\vec{v}$ a c obraz útvaru b v posunutí o $-\vec{v}$.

Aký je vzťah útvarov a a c?

Výskumná úloha 5:

Nech a je útvar a $\vec{v}$ a $\vec{w}$ sú vektory.

Označme b obraz útvaru a v posunutí o $\vec{v}$ a x obraz útvaru b v posunutí o $\vec{w}$.

Označme c obraz útvaru a v posunutí o $\vec{w}$ a y obraz útvaru c v posunutí o $\vec{v}$.

Aký je vzťah útvarov x a y?

Záleží na poradí vykonania dvoch posunutí?

2.2 Otočenie okolo počiatku súradnicovej sústavy

Príklad 16:

Kam sa zobrazí bod $(3, 1)$ v otočení okolo počiatku o uhol

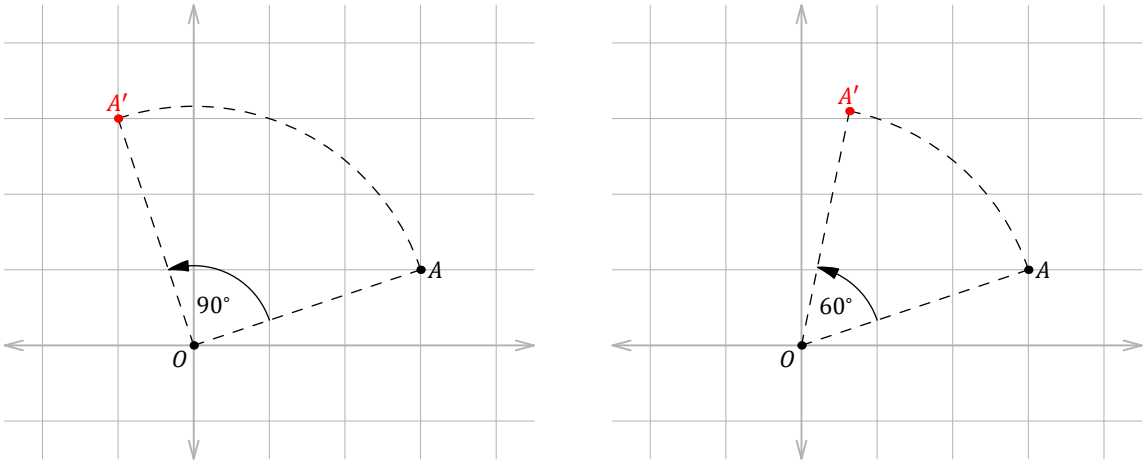
a) $90°$;

b) $60°$?

Riešenie

Počiatok označme O, vzor A a jeho obraz A'.

Zakreslime oba prípady:



Vyzerá to, že v prvom prípade $A' = (-1, 3)$. A naozaj, skalárny súčin vektorov $\overrightarrow{OA}$ a $\overrightarrow{OA'}$ je vtedy $(3, 1) \cdot (-1, 3)$ čiže 0, čo znamená, že sú kolmé, a oba majú rovnakú veľkosť $\sqrt{3^2 + 1^2}$. Táto úloha je teda riešiteľná nám doteraz známymi prostriedkami.

Žiaľ, v druhom prípade súradnice obrazu A' nie sú celé čísla. Úloha vyjadriť ich explicitne presahuje zámer i náročnosť tohto textu, takže úlohu zatiaľ splniť nevieme.

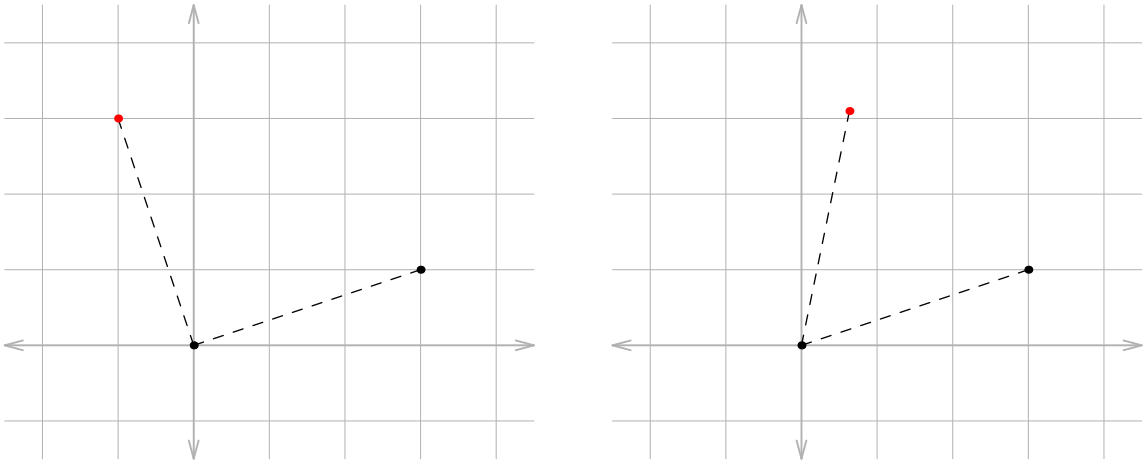
Náš ďalší postup však od toho našťastie nezávisí. Asymptote má totiž na otáčanie okolo počiatku tento jednoduchý nástroj (i keď ten v skutočnosti zrejme riešenie tejto úlohy v nejakej forme využíva):

Na otočenie okolo počiatku súradnicovej sústavy sa používa príkaz `rotate` (otoč), a to takto: Ak s je veľkosť (v kladnom smere) orientovaného uhla v stupňoch a M je množina bodov alebo jeden bod, príkaz `rotate(`s`)*`M vráti množinu bodov, resp. bod, ktorá, resp. ktorý vznikne po otočení M o s stupňov okolo počiatku.

Pomocou tohto príkladu možno prvý prípad naprogramovať takto:

```
kresliSiet(-2,4, -1,4);

pair O = (0,0);

real s = 90;
pair A = (3,1);

pair AA = rotate(s)*A;

draw(O--A,dashed);
draw(O--AA,dashed);

dot(O);
dot(A);
dot(AA,red);
```

a v druhom stačí hodnotu `s` zmeniť na `60`.



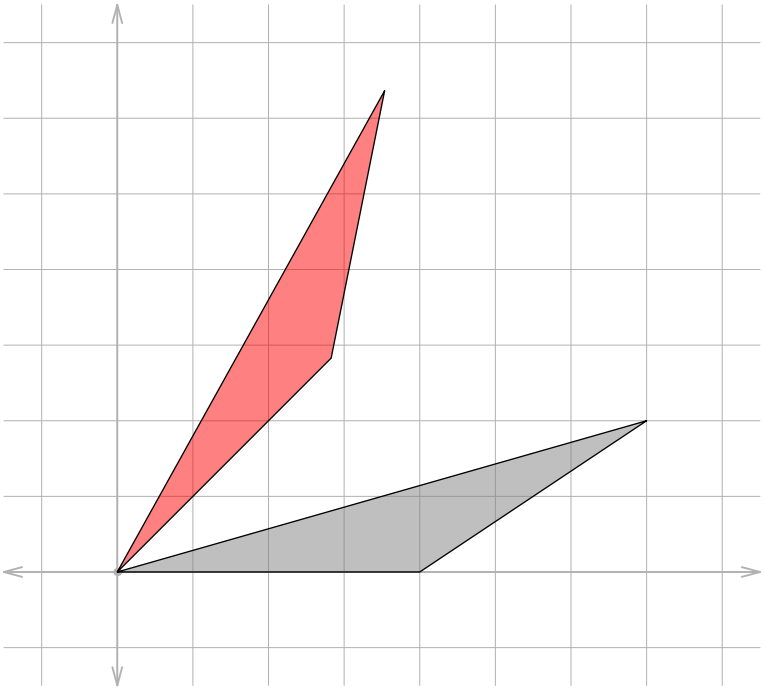
Príklad 17:

Kam sa zobrazí trojuholník s vrcholmi $(0,0)$, $(4,0)$ a $(7,2)$ v otočení o uhol $45°$ okolo počiatku?

Riešenie

Úlohu vyriešime podobne ako predchádzajúcu, len otočenie, ktoré je výsledkom príkazu `rotate`, aplikujeme tentoraz na krivku tvoriacu obvod nášho trojuholníka:

```
kresliSiet(-1,8, -1,7);

real s = 45;

path t = (0,0)--(4,0)--(7,2)--cycle;

path tt = rotate(s)*t;

fill(t,gray+opacity(.5));
draw(t);

fill(tt,red+opacity(.5));
draw(tt);
```



Výskumná úloha 6:

Nech a je útvar a s reálne číslo.

Označme b obraz útvaru a v otočení okolo počiatku o s stupňov a c obraz útvaru b v otočení okolo počiatku o $-s$ stupňov.

Aký je vzťah útvarov a a c?

Výskumná úloha 7:

Nech a je útvar a s a r reálne čísla.

Označme b obraz útvaru a v otočení okolo počiatku o s stupňov a x obraz útvaru b v otočení okolo počiatku o r stupňov.

Označme c obraz útvaru a v otočení okolo počiatku o r stupňov a y obraz útvaru c v otočení okolo počiatku o s stupňov.

Aký je vzťah útvarov x a y?

Záleží na poradí vykonania dvoch otočení okolo počiatku?

2.3 Otočenie okolo ľubovoľného bodu

Zatiaľ sme sa zaoberali len špeciálnym prípadom otočenia, a to okolo počiatku súradnicovej sústavy. Nerobili sme to však nadarmo, lebo ho s úspechom využijeme pri otočení okolo ľubovoľného bodu.

Príklad 18:

Kam sa zobrazí trojuholník s vrcholmi $(5,-5)$, $(4,-1)$ a $(6,-2)$ v otočení o uhol $100°$ okolo bodu $(5,-5)$?

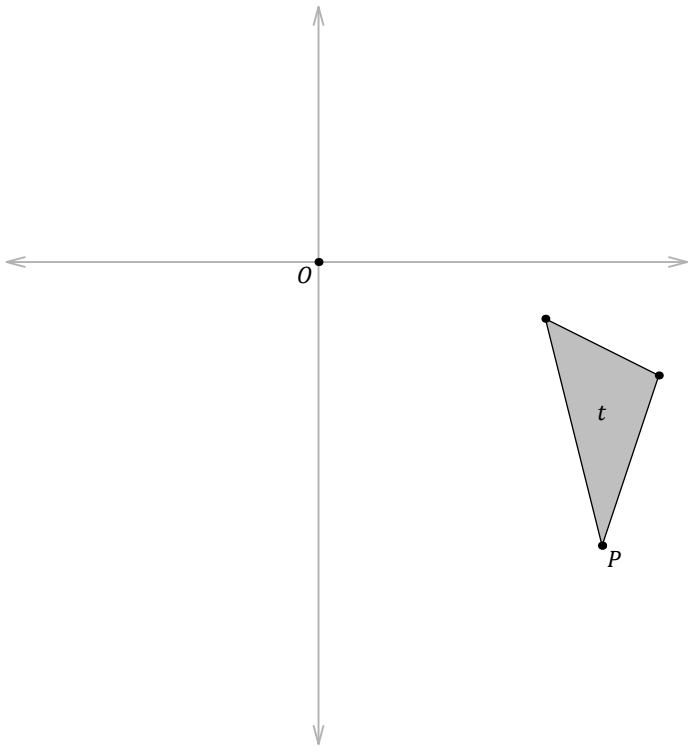
Riešenie

Najprv malé prirovnanie: Ak sa nám doma pokazí nejaké zariadenie a nie sme ho schopní svojpomocne opraviť, jednoducho ho odnesieme ho servisu, kde ho, dúfajme, dajú do poriadku a my si ho po čase odnesieme späť domov.

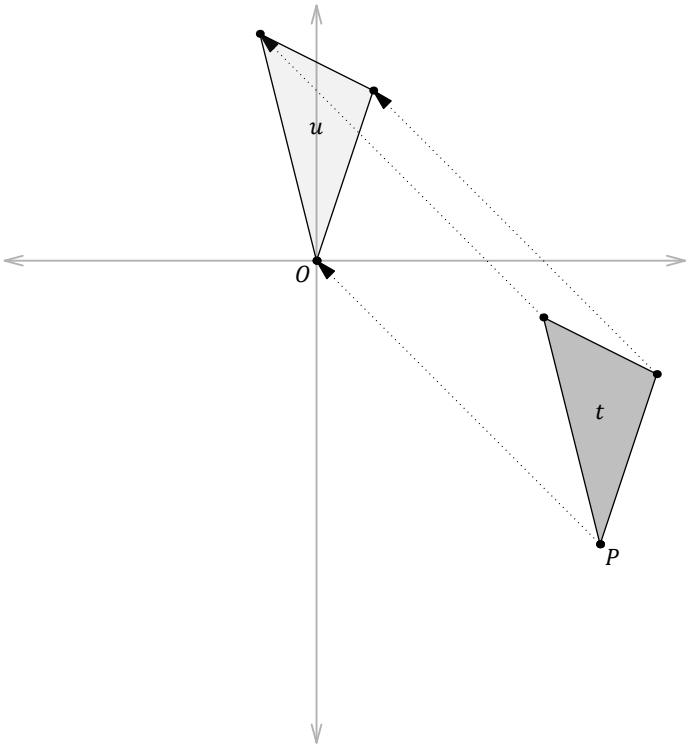
V našom prípade je „pokazeným zariadením“ nejaký útvar, jeho „oprava“ znamená otočenie o daný uhol okolo daného bodu. Keďže to (ak tento bod nie je zrovna počiatkom) nevieme (aspoň zatiaľ) urobiť, „odnesieme“ čiže posunieme ho tak, aby sa bod otočenia dostal do „servisu“ čiže do počiatku súradnicovej sústavy, podľa ktorého ho otočiť vieme. Takto posunutý útvar už môžeme nechať „opraviť“ čiže otočiť a „opravený“ ho „odnesieme domov“ čiže posunieme späť na pôvodné miesto.

Označme počiatok súradnicovej sústavy O, stred otočenia P a uhol otočenia v stupňoch s. Nakreslíme postupne všetky štyri etapy v zmysle predchádzajúceho odseku (kvôli lepšej viditeľnosti nebudeme kresliť celú sieť, len osi):

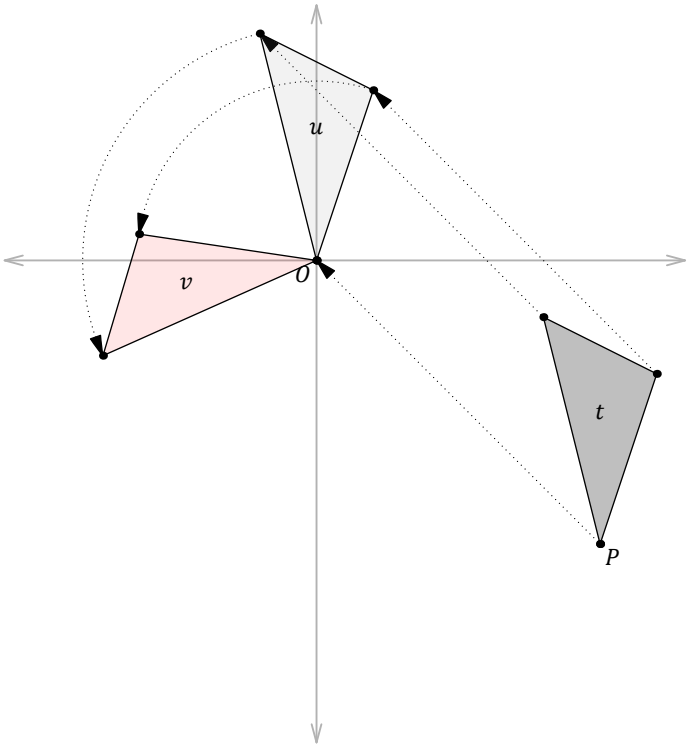
- t – pôvodný trojuholník:



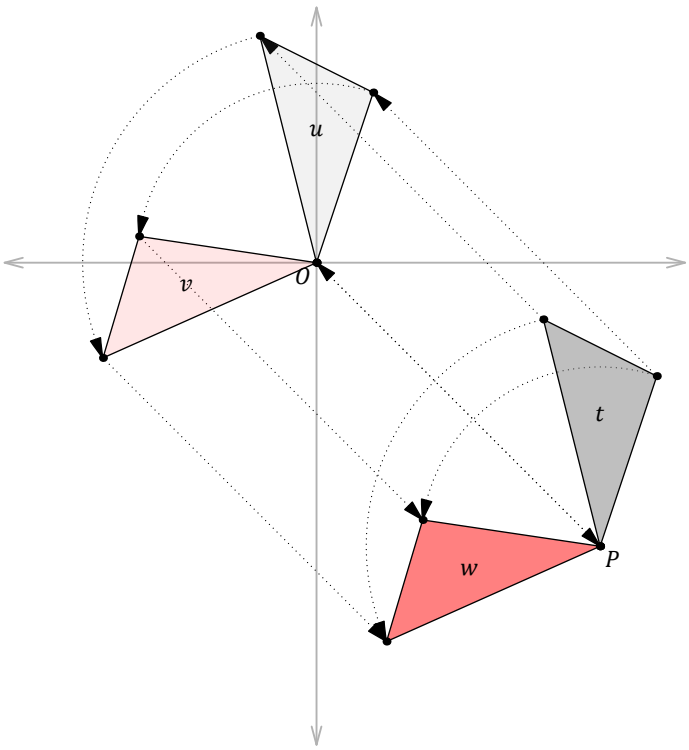
- u – trojuholník vzniknutý z t posunutím, v ktorom sa stred otočenia P dostane do počiatku O, čiže o vektor $\overrightarrow{PO}$:



- v – trojuholník vzniknutý z u otočením okolo počiatku o uhol s stupňov:

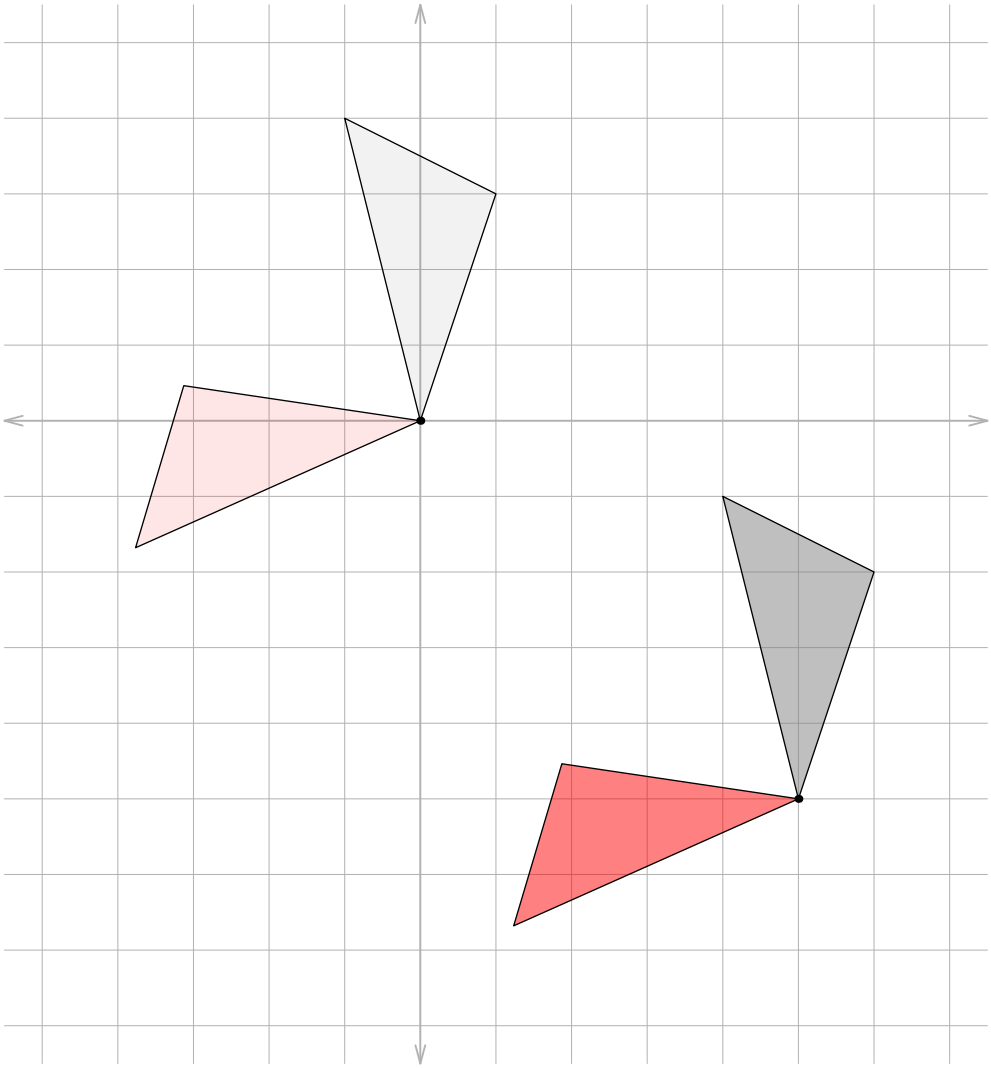


- w – trojuholník vzniknutý z v posunutím, v ktorom sa počiatok O dostane do stredu otočenia P, čiže o vektor $\overrightarrow{OP}$:



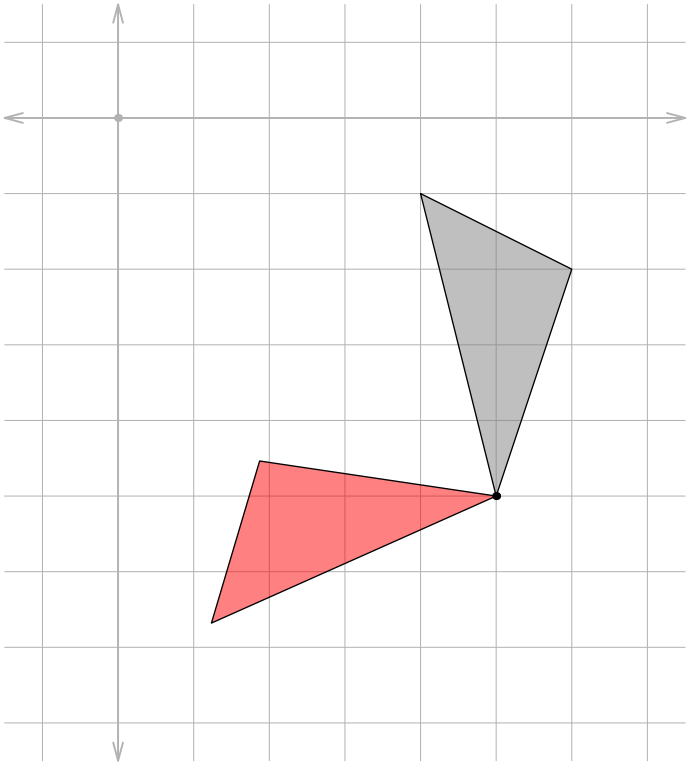
Naša konštrukcia teda môže vyzerať takto:

```
kresliSiet(-5,6, -8,4);

pair O = (0,-0);
pair P = (5,-5);
real s = 100;

pair A = (5,-5);
pair B = (4,-1);
pair C = (6,-2);

path t = A--B--C--cycle;
path u = shift(O-P)*t;
path v = rotate(s)*u;
path w = shift(P-O)*v;

fill(t,gray+opacity(.5));
draw(t);
fill(u,gray+opacity(.1));
draw(u);
fill(v,red+opacity(.1));
draw(v);
fill(w,red+opacity(.5));
draw(w);

dot(O);
dot(P);
```



Uvedomme si ešte, že pomocné medzivýsledky tejto konštrukcie u a v a ani bod 0 vykresľovať vlastne vôbec netreba. Môžeme preto zmenšiť aj sieť. Náš kód sa teda zredukuje takto:

```
kresliSiet(-1,7, -8,1);

pair O = (0,-0);
pair P = (5,-5);
real s = 100;

pair A = (5,-5);
pair B = (4,-1);
pair C = (6,-2);

path t = A--B--C--cycle;
path u = shift(O-P)*t;
path v = rotate(s)*u;
path w = shift(P-O)*v;

fill(t,gray+opacity(.5));
draw(t);

fill(w,red+opacity(.5));
draw(w);

dot(P);
```



Vieme sa dokonca vyhnúť aj deklarovaniu premenných u a v zlúčením troch riadkov

```
path u = shift(O-P)*t;
path v = rotate(s)*u;
path w = shift(P-O)*v;
```

do jediného:

```
path w = shift(O-P)*rotate(s)*shift(O-P)*t;
```

Takto zjednodušený program

```
kresliSiet(-1,7, -8,1);

pair O = (0,-0);
pair P = (5,-5);
real s = 100;

pair A = (5,-5);
pair B = (4,-1);
pair C = (6,-2);

path t = A--B--C--cycle;
path w = shift(P-O)*rotate(s)*shift(O-P)*t;

fill(t,gray+opacity(.5));
draw(t);

fill(w,red+opacity(.5));
draw(w);

dot(P);
```

generuje rovnaký obrázok. Tento prístup nás upozorňuje na možnosť vytvárať vlastné zobrazenia skladaním jednoduchších.

V zjednodušovaní programu môžeme pokročiť ešte ďalej, keď použijeme inú verziu príkazu `rotate`:

Otočenie množiny bodov M okolo bodu P o uhol s stupňov možno vyžiadať príkazom `rotate(`s`,`P`)*`M.

Za použitia tohto príkazu náš program (s rovnakým výsledkom) vyzerá takto:

```
kresliSiet(-1,7, -8,1);

pair P = (5,-5);
real s = 100;

pair A = (5,-5);
pair B = (4,-1);
pair C = (6,-2);

path t = A--B--C--cycle;
path w = rotate(s,P)*t;

fill(t,gray+opacity(.5));
draw(t);

fill(w,red+opacity(.5));
draw(w);

dot(P);
```

Výskumná úloha 8:

Nech a je útvar, P bod a s reálne číslo.

Označme b obraz útvaru a v otočení okolo P o s stupňov a c obraz útvaru b v otočení okolo P o $-s$ stupňov.

Aký je vzťah útvarov a a c?

Výskumná úloha 9:

Nech a je útvar, P bod a s a r reálne čísla.

Označme b obraz útvaru a v otočení okolo P o s stupňov a x obraz útvaru b v otočení okolo P o r stupňov.

Označme c obraz útvaru a v otočení okolo P o r stupňov a y obraz útvaru c v otočení okolo P o s stupňov.

Aký je vzťah útvarov x a y?

Záleží na poradí vykonania dvoch otočení okolo toho istého bodu?

Výskumná úloha 10:

Nech a je útvar, $\vec{v}$ vektor, P bod a s reálne číslo.

Označme b obraz útvaru a v posunutí o $\vec{v}$ a x obraz útvaru b v otočení okolo P o s stupňov.

Označme c obraz útvaru a v otočení okolo P o s stupňov a y obraz útvaru c v posunutí o $\vec{v}$.

Aký je vzťah útvarov x a y?

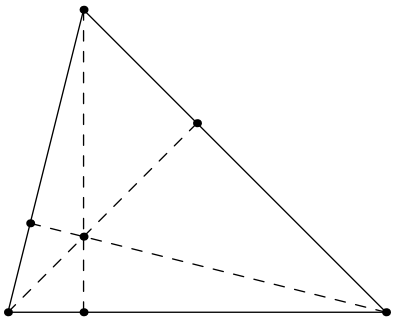
Záleží na poradí vykonania posunutia a otočenia?

3

Význačné body v trojuholníku

3.1 Výšky a ortocentrum

Pod *výškou* trojuholníka rozumieme v širšom zmysle kolmicu na priamku jeho strany prechádzajúcu jeho protiľahlým vrcholom, v užšom potom úsečku (či dokonca jej dĺžku), ktorej jedným krajným bodom je tento vrchol a druhým *päta* tejto kolmice na túto priamku čiže ich priesečník. Dá sa dokázať, že všetky tri takto vzniknuté výšky (chápané ako priamky) sa pretínajú v jedinom bode, ktorý sa nazýva *ortocentrum*.



Problém nájdenia ortocentra tak môžeme rozložiť na dva jednoduchšie podproblémy:

- nájdenie päty kolmice z daného bodu na danú priamku,
- nájdenie priesečníka dvoch priamok.

Prvý z podproblémov pritom budeme využívať hneď dvakrát.

Druhý problém rieši samotný jazyk Asymptote:

- Na skonštruovanie priesečníka rôznobežiek slúži príkaz `extension` (predĺženie), ktorého vstupmi sú štyri body, pričom prvé dva určujú prvú priamku a druhé dva druhú.
- Už názov tohto príkazu naznačuje, že bod `extension(X1,Y1, X2,Y2)` nemusí byť priesečníkom úsečiek `X1--Y1` a `X2--Y2`, ale môže ležať na ich predĺženiach.
- Zdôraznime, že predmetné priamky musia byť rôznobežné, inak pri vykonávaní programu nastane chyba.

Sústreďme sa teda na prvý problém:

Príklad 19:

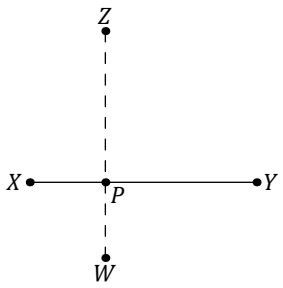
Nájdite pätu kolmice na zvolenú úsečku prechádzajúcu cez zvolený bod.

Riešenie

Označme krajné body našej úsečky X a Y a zvolený bod nech je Z.

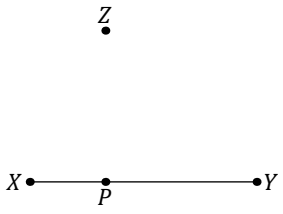
Vektor kolmý na $\overrightarrow{XY}$ získame ľahko – stačí ho otočiť (napríklad okolo začiatku) o 90°. Ak tento vektor umiestnime do bodu Z a jeho koncový bod označíme W, priamka ZW bude hľadanou kolmicou. Päta P výšky potom bude jej priesečníkom s priamkou XY.

```
pair X = (0,0);
pair Y = (3,0);
pair Z = (1,2);

pair WW = Z+rotate(90)*(X-Y);

pair P = extension(X,Y, Z,WW);

draw(X--Y);
draw(Z--WW,dashed);

dot("$X$",X,W);
dot("$Y$",Y,E);
dot("$Z$",Z,N);
dot("$W$",WW,S);
dot("$P$",P,SE);
```



A bez vykreslenia medziproduktov:

```
pair X = (0,0);
pair Y = (3,0);
pair Z = (1,2);

pair WW = Z+rotate(90)*(X-Y);
pair P = extension(X,Y, Z,WW);

draw(X--Y);

dot("$X$",X,W);
dot("$Y$",Y,E);
dot("$Z$",Z,N);

dot("$P$",P,S);
```



Aby sme mohli práve nájdené riešenie opakovane používať (máme to predsa v úmysle urobiť aspoň dvakrát), bolo by dobré, keby sme ho transformovali na vlastný príkaz. (To, že okrem príkazov ponúkaných samotným programom Asymptote môžeme vytvárať i nové, sme už určite postrehli napríklad pri použití umelého príkazu `kresliSiet`.) Najpriliehavejšie meno pre náš nový príkaz bude zrejme `pataKolmice`.

Pozrime sa na program z predchádzajúcej úlohy podrobnejšie. Má tri v podstate nezávislé časti:

- Prvá je tvorená prvými troma príkazmi a ide vlastne o prijatie vstupných hodnôt. Budú to tri vstupy nášho nového príkazu a všetky budú typu `pair`.
- V ďalších dvoch príkazoch prebieha samotný výpočet, ktorého výsledkom je hľadaná päta kolmice. Tie budú jadrom implementácie nášho príkazu. Výsledok je tiež uložený v premennej typu `pair` a bude to jeho výstup.
- Posledných päť príkazov je už „len“ vykreslenie vstupov a výsledku. Tu si treba uvedomiť, že pri programovaní sú výpočet hodnoty a jej výpis dve v podstate nesúvisiace veci. Náš príkaz sa zameriava len na tú prvú, o vykreslenie sa postará niekto iný. Túto pasáž preto ignorujeme.

- Nový vlastný *príkaz s návratovou hodnotou* s názvom p, zoznamom parametrov z, dátového typu návratovej hodnoty t a definíciou d má tvar $t\ p\ (z)\ \{d\}$.

  Pod *definíciou* alebo *telom* obvykle rozumieme sekvenciu príkazov, ktoré môžu obsahovať premenné zo zoznamu parametrov z, ale i ďalšie premenné – *lokálne* čiže deklarované v d i *globálne* čiže deklarované v celom programe.

  Definícia sa musí končiť príkazom `return` (vráť) nasledovaným výrazom dátového takého typu, ako je deklarovaná návratová hodnota. Hodnota tohto výrazu je potom výsledkom volania tohto príkazu.
- Podobne nový vlastný *príkaz bez návratovej hodnoty* s názvom p, zoznamom parametrov z a definíciou d má tvar `void` $p\ (z)\ \{d\}$ (pričom slovo `void` (prázdna) zdôrazňuje, že príkaz nemá návratovú hodnotu).
- Obvykle sú kvôli prehľadnosti tieto zápisy formátované tak, že „$t\ p\ (z)$“, resp. „`void` $p\ (z)$“ je v samostatnom riadku, v ďalšom riadku je znak `{`, potom nasleduje d rozpísané do niekoľkých riadkov, z ktorých každý je odsadený, a v poslednom riadku je znak `}`.
- i. parameter má tvar $t_i\ m_i$ alebo $t_i\ m_i = h_i$, kde t_i je jeho dátový typ, m_i jeho meno a prípadné h_i jeho predefinovaná hodnota.

Náš novovytvorený príkaz teda bude vyzerať takto:

```
pair pataKolmice (pair X, pair Y, pair Z)
{
  pair WW = Z+rotate(90)*(X-Y);
  pair P = extension(X,Y, Z,WW);
  return P;
}
```

Typ jeho návratovej hodnoty je `pair` a parametre sú `X`, `Y` a `Z`, všetky tiež typu `pair` a bez predefinovanej hodnoty. Telo je tvorené sekvenciou troch príkazov, z ktorých posledný je `return`. Sú tu deklarované lokálne premenné `WW` a `P`, ktoré sa spolu s parametrami používajú pri výpočte.

Tento príkaz môžeme prirovnať k divadelnej hre, jeho telo predstavuje jej scenár. Sú v nej tri „postavy“ `X`, `Y` a `Z` a „rekvizity“ `WW` a `P`. Každé zavolanie príkazu znamená novú inscenáciu tejto hry, musíme však pritom povedať, ktorý „herec“ bude hrať ktorú „rolu“. A tak napríklad „inscenácia“ `pataKolmice(A,B,C)` znamená, že „rolu“ `X` bude hrať „herec“ `A`, „rolu“ `Y` „herec“ `B` a „rolu“ `Z` „herec“ `C`, a pri inej „inscenácii“ `pataKolmice(C,D,D)` bude hrať „herec“ `C` „rolu“ `X` (hoci v predošlej „inscenácii“ hral inú „rolu“) a „herec“ `D` bude mať „dvojúlohu“ – bude hrať ako „rolu“ `Y`, tak „rolu“ `Z`.

Toto podobenstvo ostáva v platnosti pre ľubovoľný iný príkaz, a to i v ďalších programovacích jazykoch. Dokonca ho možno s úspechom aplikovať i na jazyk matematiky: definícia nejakého nového pojmu zodpovedá hlavičke príkazu a vymedzovaná situácia jeho telu.

Príklad 20:

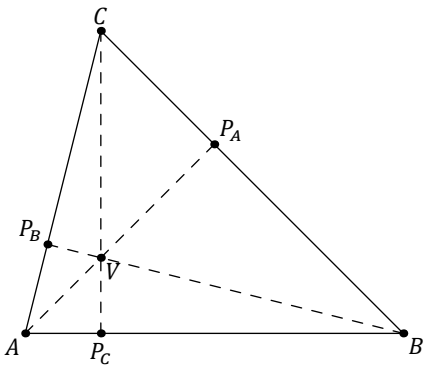
Nakreslite výšky a ortocentrum zvoleného trojuholníka.

Riešenie

Označme vrcholy zvoleného trojuholníka A, B, C, päty ich výšok postupne P_A, P_B, P_C a jeho ortocentrum V.

Keďže päty výšok sú vlastne pätami kolmíc z vrcholov na priamky protiľahlých strán, na ich konštrukciu opakovane použijeme pred chvíľou vytvorený príkaz `pataKolmice`. Náš program teda bude vyzerať takto:

```
pair pataKolmice (pair X, pair Y, pair Z)
{
  pair WW = Z+rotate(90)*(X-Y);
  pair P = extension(X,Y, Z,WW);
  return P;
}

pair A = (0,0);
pair B = (5,0);
pair C = (1,4);

pair PA = pataKolmice(B,C, A);
pair PB = pataKolmice(C,A, B);
pair PC = pataKolmice(A,B, C);

pair V = extension(A,PA, B,PB);

draw(A--B--C--cycle);

draw(A--PA,dashed);
draw(B--PB,dashed);
draw(C--PC,dashed);

dot("$A$",A,SW);
dot("$B$",B,SE);
dot("$C$",C,N);

dot("$P_A$",PA,NE);
dot("$P_B$",PB,NW);
dot("$P_C$",PC,S);

dot("$V$",V,SE);
```



Po vložení príkazu `pataKolmice` na koniec knižnice `asydef2.tex` bude náš program už bez neho:

```
pair A = (0,0);
pair B = (5,0);
pair C = (1,4);

pair PA = pataKolmice(B,C, A);
pair PB = pataKolmice(C,A, B);
pair PC = pataKolmice(A,B, C);

pair V = extension(A,PA, B,PB);

draw(A--B--C--cycle);

draw(A--PA,dashed);
draw(B--PB,dashed);
draw(C--PC,dashed);

dot("$A$",A,SW);
dot("$B$",B,SE);
dot("$C$",C,N);

dot("$P_A$",PA,NE);
dot("$P_B$",PB,NW);
dot("$P_C$",PC,S);

dot("$V$",V,SE);
```

Aj nájdenie ortocentra môžeme pretvoriť na samostatný príkaz, ktorý tiež pridáme do knižnice `asydef2.tex`, avšak až za príkaz `pataKolmice`, pretože ho využíva:

```
pair ortocentrum (pair X, pair Y, pair Z)
{
  pair PX = pataKolmice(Y,Z, X);
  pair PY = pataKolmice(X,Z, Y);
  pair V = extension(X,PX, Y,PY);
  return V;
}
```

Výskumná úloha 11:

Popíšte polohu ortocentra
- ostrouhlého,
- tupouhlého,
- pravouhlého

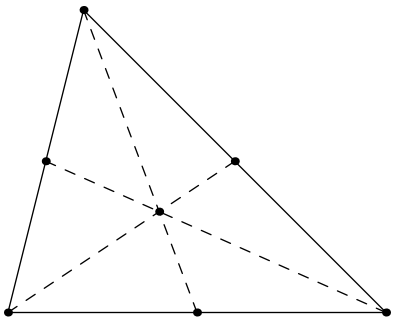
trojuholníka.

Výskumná úloha 12:

Nech V je ortocentrum nepravouhlého trojuholníka ABC. Zistite, aký vzťah má bod C k trojuholníku ABV.

3.2 Ťažnice a ťažisko

Pripomeňme, že *ťažnica* trojuholníka je úsečka, ktorá spája vrchol trojuholníka so stredom protiľahlej strany, a že všetky tri ťažnice prechádzajú tým istým bodom, ktorý nazývame *ťažisko*.

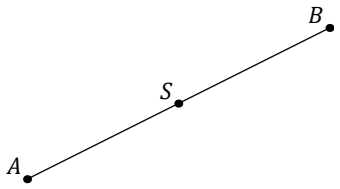


Príklad 21:

Zvoľte si ľubovoľnú úsečku a nakreslite jej stred.

Riešenie

Označme krajné body našej úsečky A a B, jej stred nech je S.



Vektor $\overrightarrow{AS}$ má teda rovnaký smer a orientáciu ako vektor $\overrightarrow{AB}$ a má voči nemu polovičnú veľkosť. Môžeme preto písať:

$$\overrightarrow{AS} = \frac{1}{2}\overrightarrow{AB}.$$

Využitím vzťahu $\overrightarrow{XY} = Y - X$ a ďalšími úpravami dostávame

$$S - A = \frac{1}{2}(B - A),$$

$$S = \frac{1}{2}(B - A) + A,$$

$$S = \frac{1}{2}B - \frac{1}{2}A + A,$$

$$S = \frac{1}{2}B + \frac{1}{2}A$$

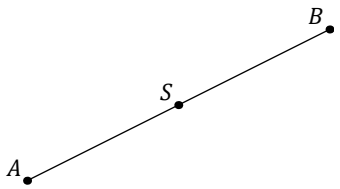
alebo krajšie

$$S = \frac{1}{2}A + \frac{1}{2}B,$$

prípadne vo forme aritmetického priemeru

$$S = \frac{A + B}{2}.$$

```
pair A = (0,0);
pair B = (4,2);

pair SS = 1/2*A+1/2*B;

draw(A--B);

dot("$A$",A,NW);
dot("$B$",B,NW);
dot("$S$",SS,NW);
```



- Bod krivky získame aplikáciou príkazu `point` (bod), ktorého prvým vstupom je táto krivka a druhým tzv. „index“ tohto jej bodu (v programovom kóde zapisovaný, samozrejme, príslušným fontom).
- V prípade úsečky sú jej body indexované rovnomerne reálnymi číslami od 0 do 1, pričom 0 je index bodu uvedeného ako prvý.

  Čiže napríklad `point(A--B,0)` je bod `A`, `point(A--B,1)` je bod `B`, `point(A--B,1/2)` je stred úsečky `A--B` a povedzme `point(A--B,2/3)` bod úsečky `A--B`, ktorý ju delí na časti v pomere $2:1$, pričom dlhšia je pri bode `A`.
- V prípade lomenej čiary tvorenej n úsečkami sú jej body indexované reálnymi číslami 0 až n, pričom bod s indexom $i+x$, kde i je prirodzené číslo od 0 do $n-1$ a x je reálne číslo od 0 do 1, je x bod $(i+1)$. úsečky.

  Čiže napríklad `point(A--B--C--D,0)` je bod `A`, `point(A--B--C--D,2)` je bod `C` a `point(A--B,2.5)` je stred úsečky `C--D`.
- Body kružnice sú indexované rovnomerne číslami 0 až 4.

- Stredný bod krivky (pri rôznych krivkách môže byť tento pojem definovaný rôzne) možno získať prostredníctvom príkazu `midpoint` (stredný bod), ktorého jediným vstupom je dotyčná krivka.
- V prípade úsečky je jej stredným bodom jej stred, takže ak sú jej krajné body `X` a `Y` (čiže ona je `X--Y`), jej stred získame výrazom `midpoint(X--Y)`.

Riadok

```
pair SS = 1/2*A+1/2*B;
```

nášho programu tak môžeme ekvivalentne nahradiť riadkom

```
pair SS = point(A--B,1/2);
```

alebo šikovnejšie

```
pair SS = midpoint(A--B);
```

Výskumná úloha 13:

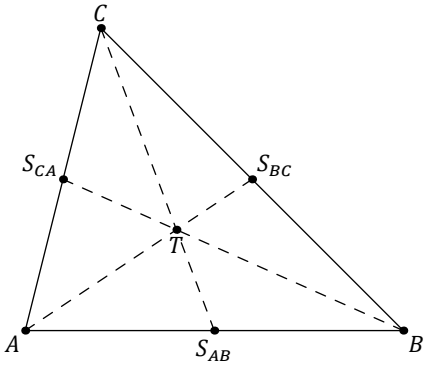
Zistite, kedy je stred úsečky s krajnými mrežovými bodmi tiež mrežovým bodom.

Príklad 22:

Zvoľte si ľubovoľný trojuholník a nakreslite jeho ťažnice a ťažisko.

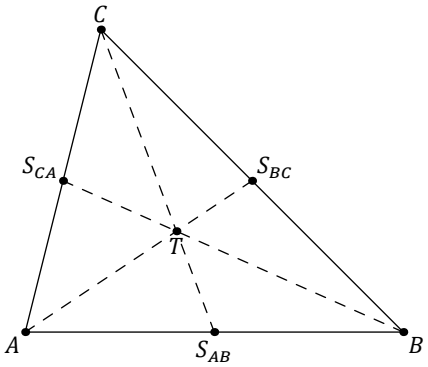
Riešenie

Označme vrcholy zvoleného trojuholníka A, B, C, stredy jeho strán AB, BC, CA postupne S_{AB}, S_{BC}, S_{CA} a jeho ťažisko T.



Keďže stred úsečky už vieme zostrojiť, s ťažnicami nebude problém. Vyberieme si teda dve z nich a ťažisko bude ich priesečníkom:

```
pair A = (0,0);
pair B = (5,0);
pair C = (1,4);

pair SAB = midpoint(A--B);
pair SBC = midpoint(B--C);
pair SCA = midpoint(C--A);

pair T = extension(A,SBC, B,SCA);

draw(A--B--C--cycle);

draw(A--SBC,dashed);
draw(B--SCA,dashed);
draw(C--SAB,dashed);

dot("$A$",A,SW);
dot("$B$",B,SE);
dot("$C$",C,N);

dot("$S_{AB}$",SAB,S);
dot("$S_{BC}$",SBC,NE);
dot("$S_{CA}$",SCA,NW);

dot("$T$",T,S);
```



Aj na nájdenie ťažiska môžeme vytvoriť vlastný príkaz a odložiť ho do knižnice:

```
pair tazisko (pair X, pair Y, pair Z)
{
  pair SXZ = midpoint(X--Z);
  pair SYZ = midpoint(Y--Z);
  pair T = extension(X,SYZ, Y,SXZ);
  return T;
}
```

Výskumná úloha 14:

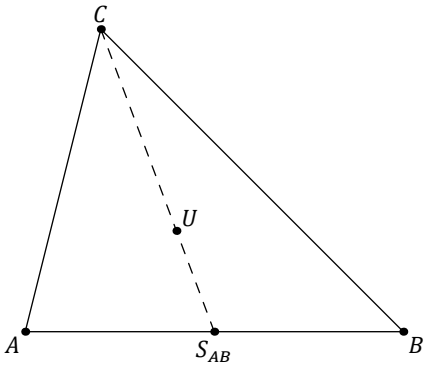
Zistite, kedy je ťažisko trojuholníka s vrcholmi v mrežových bodoch tiež mrežovým bodom.

Príklad 23:

Overte, že ťažisko delí ťažnicu v pomere $2 : 1$.

Riešenie

Použime označenie z predchádzajúcej úlohy. Bez ujmy na všeobecnosti uvažujme ťažnicu CS_{AB}. Nech U je bod, ktorý delí túto ťažnicu v pomere $2 : 1$, pričom väčšia časť je pri vrchole C.



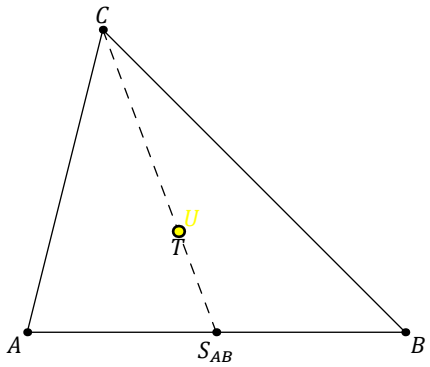
Platí teda

$$\overrightarrow{CU} = \frac{2}{3}\overrightarrow{CS_{AB}},$$

$$U - C = \frac{2}{3}(S_{AB} - C),$$

$$U = C + \frac{2}{3}(S_{AB} - C).$$

```
pair A = (0,0);
pair B = (5,0);
pair C = (1,4);

pair T = tazisko(A,B,C);

pair SAB = midpoint(A--B);
pair U = C+2/3*(SAB-C);

draw(A--B--C--cycle);

draw(C--SAB,dashed);

dot("$A$",A,SW);
dot("$B$",B,SE);
dot("$C$",C,N);

dot("$S_{AB}$",SAB,S);

dot("$T$",T,S,linewidth(5));
dot("$U$",U,NE,yellow);
```



Ako vidieť, body U a T teda (aspoň v tomto prípade) splývajú.

Samozrejme, len na základe takéhoto optického pozorovania nemôžeme tvrdiť, že to platí vo všeobecnosti. Pokračujme v úpravách:

$$U = C + \frac{2}{3}(S_{AB} - C) = C + \frac{2}{3}S_{AB} - \frac{2}{3}C = \frac{2}{3}S_{AB} + \frac{1}{3}C =$$
$$= \frac{2}{3}\left(\frac{1}{2}A + \frac{1}{2}B\right) + \frac{1}{3}C = \frac{1}{3}A + \frac{1}{3}B + \frac{1}{3}C = \frac{1}{3}(A + B + C).$$

Ako vidieť, záverečný výraz je symetrický voči bodom A, B, C. To však znamená, že ten istý bod by sme dostali, aj keby sme uvažovali ľubovoľnú inú ťažnicu. Tento bod teda leží na všetkých troch ťažniciach, takže je to naozaj ťažisko.

Ako vedľajší produkt predchádzajúcich úvah sme dostali, že ťažisko ľubovoľného trojuholníka ABC je

$$\frac{1}{3}(A + B + C).$$

To nám umožňuje zjednodušiť a zestetičtiť implementáciu nášho (už uskladneného) príkazu `tazisko` takto:

```
pair tazisko (pair X, pair Y, pair Z)
{
  return 1/3*(X+Y+Z);
}
```

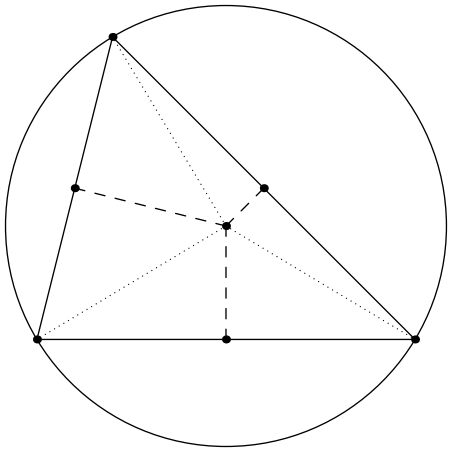
Uvedomme si, že takáto reimplementácia nemá (a ani nesmie mať) žiaden vplyv na výpočty, ktoré tento príkaz používajú. On je pre ne akousi „čiernou skrinkou“ – to, ako sa v ňom hľadá a nájde správna hodnota, volajúceho absolútne nezaujíma.

Výskumná úloha 15:

Nech T je ťažisko trojuholníka ABC a D, E, F postupne ťažiská trojuholníkov ABT, BCT, CAT. Aký je vzťah trojuholníka DEF a bodu T?

3.3 Opísaná kružnica

Kružnica opísaná trojuholníku je jediná kružnica, ktorá prechádza jeho všetkými tromi vrcholmi. To znamená, že jej stred musí byť od každého z nich vzdialený rovnako, a teda musí ležať na každej z osí jeho strán. Od tohto faktu sa odvíja i jeho konštrukcia.



Príklad 24:

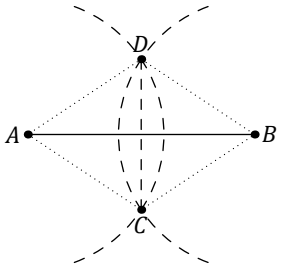
Zvoľte si ľubovoľnú úsečku a nakreslite jej os.

Riešenie

Úlohu vyriešime dvoma spôsobmi, každý z nich bude pre nás svojím spôsobom poučný.

1 Najprv napodobníme klasickú konštrukciu, pri ktorej sa zostroja rovnako veľké kružnice (prípadne iba ich relevantné oblúky) so stredmi v oboch krajných bodoch a s ľubovoľným polomerom veľkosti aspoň pol dĺžky úsečky. Os úsečky je potom priamka, ktorá prechádza priesečníkmi týchto kružníc.

Ak krajné body našej úsečky označíme A a B a priesečníky (oblúkov) kružníc C a D, všetky štyri úsečky AC, BC, AD, BD majú rovnakú dĺžku (rovnú zvolenému spoločnému polomeru oboch kružníc), čo znamená, že $ABCD$ je kosoštvorec. A keďže uhlopriečky kosoštvorca sú navzájom kolmé a rozpoľujú sa, priamka CD je osou úsečky AB, čo zdôvodňuje správnosť tejto konštrukcie.



Na to, aby sme zostrojili kružnice, musíme poznať ich spoločný polomer (ich stredy sú A a B). Ten má byť aspoň polovicou dĺžky úsečky AB, tak nech je kvôli jednoduchosti priamo rovný jej dĺžke. Potrebujeme teda vedieť vyčísliť dĺžku úsečky.

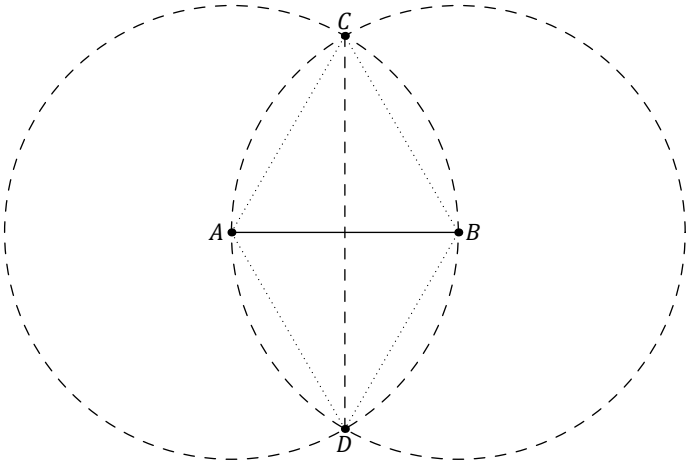
- Na určenie vzdialenosti bodu od počiatku súradnicovej sústavy sa používa príkaz `abs` (skratka pre „absolute value“ čiže absolútnu hodnotu), ktorého vstupom je tento bod.
- Tento príkaz možno použiť aj na určenie dĺžky úsečky, a to tak, že ju chápeme ako vektor a ako vstup použijeme koncový bod jeho základnej polohy. Dĺžka úsečky s krajnými bodmi `X` a `Y` je teda `abs(Y-X)` (alebo rovnako dobre `abs(X-Y)`).

Kružnice samotné však nestačia, treba vedieť nájsť ich priesečníky.

- Príkaz `intersectionpoints` (body prieniku), ktorého dvoma vstupmi sú krivky, vracia príslušne dlhý zoznam ich priesečníkov (číslovaný od nuly).
- Ak sú `k1` a `k2` krivky a `i` je index menší než dĺžka tohto zoznamu, `intersectionpoints(k1,k2)[i]` znamená `i`. priesečník týchto dvoch kriviek (rátaný pozdĺž prvej z nich). Ak však index `i` nie je menší než je dĺžka zoznamu, program sa skončí s chybou.

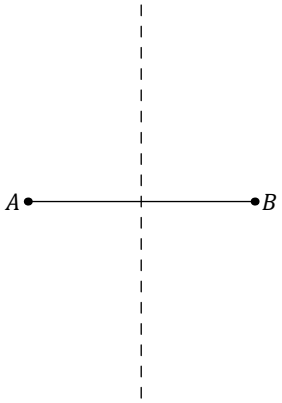
Naša konštrukcia teda môže vyzerať takto:

```
pair A = (0,0);
pair B = (3,0);

real r = abs(A-B);

path kA = circle(A,r);
path kB = circle(B,r);

pair C = intersectionpoints(kA,kB)[0];
pair D = intersectionpoints(kA,kB)[1];

draw(A--B);

draw(C--D,dashed);
draw(A--C--B--D--cycle,dotted);

draw(kA,dashed);
draw(kB,dashed);

dot("$A$",A,W);
dot("$B$",B,E);
dot("$C$",C,S);
dot("$D$",D,N);
```



A bez pomocných čiar a označení:

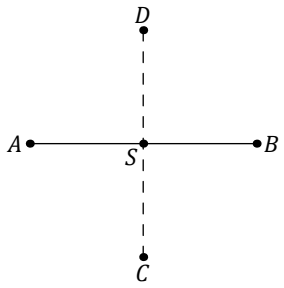
```
pair A = (0,0);
pair B = (3,0);

real r = abs(A-B);

path kA = circle(A,r);
path kB = circle(B,r);

pair C = intersectionpoints(kA,kB)[0];
pair D = intersectionpoints(kA,kB)[1];

draw(A--B);

draw(C--D,dashed);

dot("$A$",A,W);
dot("$B$",B,E);
```



2 Alternatívne riešenie využíva fakt, že os úsečky je priamka na ňu kolmá a prechádzajúca jej stredom. Stred úsečky už zostrojiť vieme, ostáva konštrukcia kolmice na danú úsečku cez jej daný bod. Tú získame jednoducho – otočením tejto úsečky okolo tohto bodu o 90°.

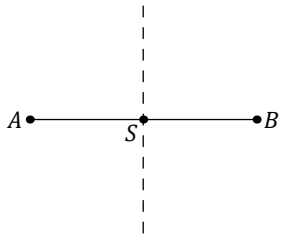
Označme krajné body našej úsečky opäť A a B, jej stred S. Jej osou bude priamka CD, kde body C a D vzniknú otočením bodov A, resp. B o 90° okolo jej stredu S.

```
pair A = (0,0);
pair B = (3,0);

pair SS = midpoint(A--B);

pair C = rotate(90,SS)*A;
pair D = rotate(90,SS)*B;

draw(A--B);

draw(C--D,dashed);

dot("$A$",A,W);
dot("$B$",B,E);
dot("$C$",C,S);
dot("$D$",D,N);
dot("$S$",SS,SW);
```



A ešte bez vykreslenia pomôcok:

```
pair A = (0,0);
pair B = (3,0);

pair SS = midpoint(A--B);

pair C = rotate(90,SS)*A;
pair D = rotate(90,SS)*B;

draw(A--B);

draw(C--D,dashed);

dot("$A$",A,W);
dot("$B$",B,E);
dot("$S$",SS,SW);
```



Príklad 25:

Zvoľte si ľubovoľný trojuholník a nakreslite jemu opísanú kružnicu.

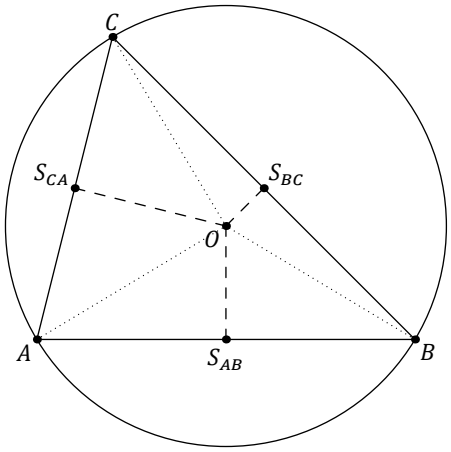
Riešenie

Označme vrcholy zvoleného trojuholníka A, B, C, stredy jeho strán AB, BC, CA postupne S_{AB}, S_{BC}, S_{CA} a stred jemu opísanej kružnice O.

Bod O je priesečníkom vybranej dvojice osí strán. Na každej potrebujeme dva body. Jeden z nich je jej stred, druhý získame konštrukciou z predchádzajúceho príkladu.

Keď už budeme mať stred hľadanej kružnice, jej polomer získame ľahko – je to jeho vzdialenosť od ktoréhokoľvek vrcholu.

```
pair A = (0,0);
pair B = (5,0);
pair C = (1,4);

pair SAB = midpoint(A--B);
pair SBC = midpoint(B--C);
pair SCA = midpoint(C--A);

pair O = extension(SAB,rotate(90,SAB)*B, SBC,rotate(90,SBC)*C);

path k = circle(O,abs(O-A));

draw(A--B--C--cycle);

draw(k);

draw(A--O,dotted);
draw(B--O,dotted);
draw(C--O,dotted);

draw(SAB--O,dashed);
draw(SBC--O,dashed);
draw(SCA--O,dashed);

dot("$A$",A,SW);
dot("$B$",B,SE);
dot("$C$",C,N);

dot("$S_{AB}$",SAB,S);
dot("$S_{BC}$",SBC,NE);
dot("$S_{CA}$",SCA,NW);

dot("$O$",O,SW);
```



Aj tu predchádzajúce riešenie prerobíme na príkaz:

```
pair stredOpisanejKruznice (pair X, pair Y, pair Z)
{
  pair SXY = midpoint(X--Y);
  pair SYZ = midpoint(Y--Z);
  pair O = extension(SXY,rotate(90,SXY)*Y, SYZ,rotate(90,SYZ)*Z);
  return O;
}
```

A môžeme pridať aj príkaz na nakreslenie samotnej opísanej kružnice, výsledok však bude mať typ `path`:

```
path opisanaKruznica (pair X, pair Y, pair Z)
{
  pair O = stredOpisanejKruznice(X,Y,Z);
  path k = circle(O,abs(O-X));
  return k;
}
```

Výskumná úloha 16:

Popíšte polohu stredu kružnice opísanej

- ostrouhlému,
- tupouhlému,
- pravouhlému

trojuholníku.

Výskumná úloha 17:

Vnútri strán BC, CA, AB trojuholníka ABC sú postupne body D, E, F. Aký vzťah majú kružnice opísané trojuholníkom EAF, FBD, DCE? Ako je to v prípade, keď D, E, F sú stredy príslušných strán?

Výskumná úloha 18:

Nech P, Q, R sú päty výšok nepravouhlého trojuholníka ABC postupne z vrcholov A, B, C. Aký je vzťah piat kolmíc z bodu P na priamky AB a AC, z bodu Q na priamky BC a BA a z bodu R na priamky CA a CB?

Výskumná úloha 19:

Nech V je ortocentrum trojuholníka ABC a D priesečník jemu opísanej kružnice s pätou výšky z vrcholu A na stranu BC. Aký je vzťah bodov V a D?

Výskumná úloha 20:

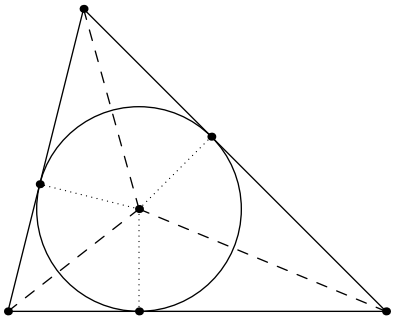
Nech V je ortocentrum trojuholníka ABC. Aký je vzťah kružníc opísaných trojuholníkom ABC a ABV?

Výskumná úloha 21:

Nech O je stred kružnice opísanej trojuholníku ABC. Nech S_{AB}, S_{BC}, S_{CA} sú postupne stredy úsečiek AB, BC, CA. Aký je vzťah bodu O a trojuholníka $S_{AB}S_{BC}S_{CA}$?

3.4 Vpísaná kružnica

Kružnica vpísaná do trojuholníka je jediná kružnica, ktorá sa dotýka všetkých jeho strán. To znamená, že jej stred musí byť od každej z nich vzdialený rovnako, a teda musí ležať na každej z osí jeho uhlov. Toto pozorovanie využijeme pri jeho konštrukcii.



Keďže klasická konštrukcia osi uhla neobsahuje z hľadiska Asymptote žiadne nové myšlienky, nechávame ju len ako (vcelku užitočné) cvičenie:

Úloha 5:

Zvoľte si uhol a napodobnite klasickú konštrukciu jeho osi.

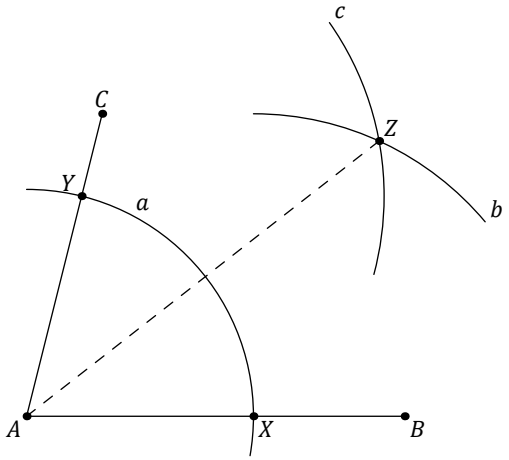
Sústredíme sa len na alternatívne riešenie:

Príklad 26:

Zvoľte si uhol a nakreslite jeho os.

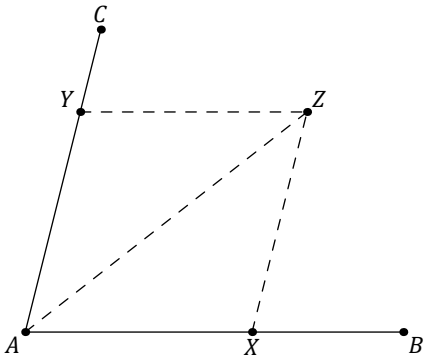
Riešenie

Odrazme sa od riešenia predchádzajúcej úlohy s nasledujúcim označením:



Keďže kružnice oblúkov b a c majú podľa konštrukcie rovnaké polomery, útvar $AXZY$ je deltoid súmerný podľa svojej uhlopriečky AZ, a tá teda naozaj rozpoľuje jeho uhol XAY čiže BAC.

Zhodné polomery oblúkov b a c zatiaľ nemajú žiaden vzťah k polomeru oblúka a, nič nám teda nebráni zvoliť všetky tri rovnaké. Deltoid $AXZY$ sa potom stane kosoštvorcom:

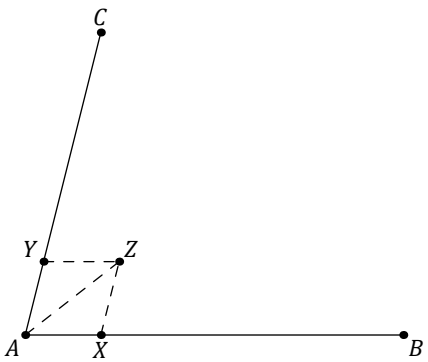


Potom platí $\overrightarrow{XZ} = \overrightarrow{AY}$, čiže $Z = X + (Y - A)$, na konštrukciu bodu Z teda nepotrebujeme konštruovať oblúky b a c.

Keďže na dĺžke strany pomocného kosoštvorca nezáleží, zvolíme pre ňu hodnotu 1. Body X a Y by sme vedeli skonštruovať ako priesečníky jednotkovej kružnice so stredom v A s ramenami uhla, použijeme však túto pomôcku:

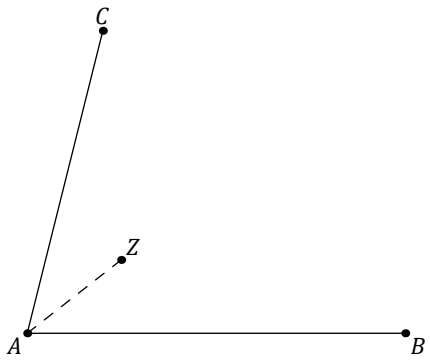
Na zostrojenie jednotkového vektora v smere daného vektora slúži príkaz `dir` („direction“, smer), ktorého vstupom je koncový bod tohto vektora.

```
pair A = (0,0);
pair B = (5,0);
pair C = (1,4);

pair X = A+dir(B-A);
pair Y = A+dir(C-A);

pair Z = X+Y-A;

draw(B--A--C);
draw(X--Z--Y,dashed);
draw(A--Z,dashed);

dot("$A$",A,SW);
dot("$B$",B,SE);
dot("$C$",C,N);

dot("$X$",X,S);
dot("$Y$",Y,NW);
dot("$Z$",Z,NE);
```



Zredukujme vykreslenie len na dôležité veci:

```
pair A = (0,0);
pair B = (5,0);
pair C = (1,4);

pair X = A+dir(B-A);
pair Y = A+dir(C-A);

pair Z = X+Y-A;

draw(B--A--C);
draw(A--Z,dashed);

dot("$A$",A,SW);
dot("$B$",B,SE);
dot("$C$",C,N);

dot("$Z$",Z,NE);
```



Podrobnejší pohľad na tento kód nám ukáže, že deklarovať premenné X a Y je zbytočné, stačí riadky

```
pair X = A+dir(B-A);
pair Y = A+dir(C-A);
pair Z = X+Y-A;
```

nahradiť jediným

```
pair Z = (A+dir(B-A))+(A+dir(C-A))-A;
```

alebo po úprave výrazu na pravej strane

```
pair Z = A+dir(B-A)+dir(C-A);
```

Dostávame tak finálne riešenie:

```
pair A = (0,0);
pair B = (5,0);
pair C = (1,4);

pair Z = A+dir(B-A)+dir(C-A);

draw(B--A--C);
draw(A--Z,dashed);

dot("$A$",A,SW);
dot("$B$",B,SE);
dot("$C$",C,N);

dot("$Z$",Z,NE);
```

Príklad 27:

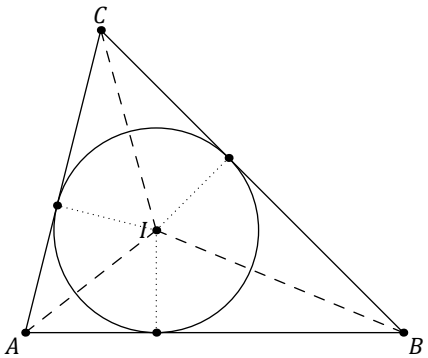
Zvoľte si ľubovoľný trojuholník a nakreslite do neho vpísanú kružnicu.

Riešenie

Označme vrcholy zvoleného trojuholníka A, B, C a stred do neho vpísanej kružnice I. Ten leží na osiach uhlov (vyberieme si dva z nich), ktoré už skonštruovať vieme.

Na zostrojenie kružnice však potrebujeme vedieť jej polomer, ktorý je rovný vzdialenosti bodu I od päty jeho výšky na ľubovoľnú stranu. Tú tiež zostrojiť vieme.

```
pair A = (0,0);
pair B = (5,0);
pair C = (1,4);

pair I = extension(A,A+dir(B-A)+dir(C-A), B,B+dir(A-B)+dir(C-B));

pair P = pataKolmice(A,B, I);
pair Q = pataKolmice(B,C, I);
pair R = pataKolmice(C,A, I);

path k = circle(I,abs(I-P));

draw(A--B--C--cycle);

draw(I--P,dotted);
draw(I--Q,dotted);
draw(I--R,dotted);

draw(I--A,dashed);
draw(I--B,dashed);
draw(I--C,dashed);

draw(k);

dot("$A$",A,SW);
dot("$B$",B,SE);
dot("$C$",C,N);

dot(P);
dot(Q);
dot(R);

dot("$I$",I,SW);
```



Tradične vyrobíme aj samostatné príkazy, ktoré vložíme do knižnice `asydef2.tex`:

```
pair stredVpisanejKruznice (pair X, pair Y, pair Z)
{
  return extension(X,X+dir(Y-X)+dir(Z-X), Y,Y+dir(X-Y)+dir(Z-Y));
}

path vpisanaKruznica (pair X, pair Y, pair Z)
{
  pair I = stredVpisanejKruznice(X,Y,Z);
  pair P = pataKolmice(X,Y, I);
  path k = circle(I,abs(I-P));
  return k;
}
```

Výskumná úloha 22:

Nech I je stred kružnice vpísanej trojuholníku ABC a D je priesečník kružnice jemu opísanej s osou uhla ACB rôzny od C. Aký je vzťah bodu D a trojuholníka ABI?

Výskumná úloha 23:

Nech P, Q, R sú päty výšok nepravouhlého trojuholníka ABC. Nech I je stred kružnice vpísanej trojuholníku PQR. Aký je vzťah bodu I a trojuholníka ABC?

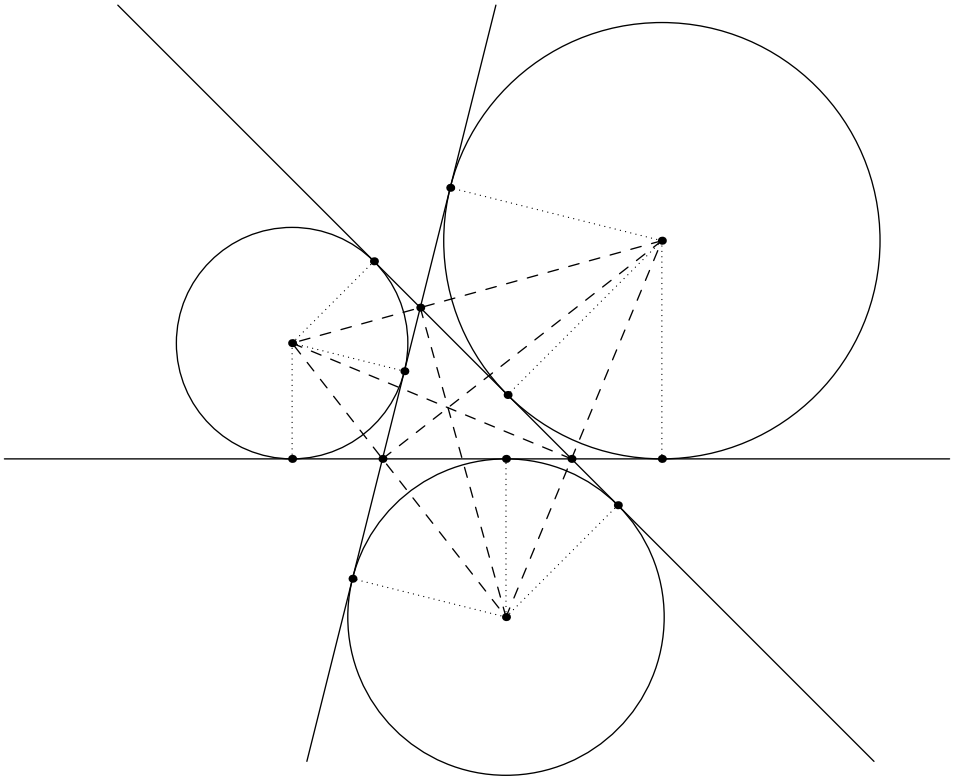
Výskumná úloha 24:

Nech I je stred kružnice vpísanej do trojuholníka ABC. Aký je vzťah priesečníkov osí úsečiek AI, BI, CI k trojuholníku ABC?

3.5 Pripísané kružnice

Kružnica *pripísaná* do trojuholníka ABC k jeho strane AB je jediná kružnica, ktorá sa dotýka strany AB a polpriamok opačných k polpriamkam AC a BC.

Na obrázku sú znázornené všetky tri pripísané kružnice k danému trojuholníku:



Ako sa dá vidieť, ich stredy tiež ležia na osiach uhlov určených priamkami strán trojuholníka, čo dáva istý návod na ich konštrukciu:

Úloha 6:

Minimalistickou úpravou príkazu na nájdenie stredu kružnice vpísanej do daného trojuholníka vytvorte príkaz na nájdenie stredu kružnice pripísanej k jeho zvolenej strane.

Úloha 7:

Minimalistickou úpravou príkazu na nájdenie kružnice vpísanej do daného trojuholníka vytvorte príkaz na nájdenie kružnice pripísanej k jeho zvolenej strane.

Úloha 8:

Pomocou príkazu z predchádzajúcej úlohy zostrojte pre zvolený trojuholník všetky tri k nemu pripísané kružnice.

Výskumná úloha 25:

Nech I je stred kružnice vpísanej do daného trojuholníka a J, K, L stredy kružníc k nemu pripísaných. Aký je vzťah bodu I a trojuholníka JKL?

Výskumná úloha 26:

Nech O je stred kružnice opísanej trojuholníku ABC, I stred kružnice doň vpísanej a J, K, L stredy kružníc k nemu pripísaných. Nech S je stred kružnice opísanej trojuholníku JKL. Aký je vzťah bodov I, O a S?

3.6 Feuerbachova kružnica

Feuerbachovou kružnicou daného trojuholníka nazývame kružnicu, ktorá prechádza stredmi jeho strán.

Úloha 9:

Nakreslite Feuerbachovu kružnicu zvoleného trojuholníka.

Úloha 10:

Vytvorte príkaz na nájdenie stredu Feuerbachovej kružnice daného trojuholníka.

Úloha 11:

Vytvorte príkaz na nájdenie Feuerbachovej kružnice daného trojuholníka.

Výskumná úloha 27:

Zistite vzájomnú polohu Feuerbachovej kružnice daného trojuholníka a piat jeho výšok.

Výskumná úloha 28:

Nech ABC je trojuholník a V jeho ortocentrum. Zistite vzájomnú polohu Feuerbachovej kružnice trojuholníka ABC a stredov úsečiek VA, VB, VC.

Výskumná úloha 29:

Zistite vzájomný vzťah Feuerbachovej kružnice daného trojuholníka a kružnice jemu opísanej.

Výskumná úloha 30:

Zistite vzájomný vzťah Feuerbachovej kružnice daného trojuholníka ku kružnici doňho vpísanej a ku kružniciam k nemu pripísaným.

Výskumná úloha 31:

Nech J, K, L sú stredy kružníc pripísaných k trojuholníku ABC. Nech f je Feuerbachova kružnica trojuholníka JKL. Aký je vzťah kružnice f k trojuholníku ABC?

Výskumná úloha 32:

Zistite vzájomnú polohu ortocentra, stredu jeho opísanej kružnice a stredu jeho Feuerbachovej kružnice.

3.7 Eulerova priamka

Pod *Eulerovou priamkou* daného nerovnostranného trojuholníka rozumieme priamku prechádzajúcu jeho ortocentrom a stredom jemu opísanej kružnice.

Úloha 12:

Nakreslite Eulerovu priamku zvoleného trojuholníka.

Výskumná úloha 33:

Zistite vzájomnú polohu Eulerovej priamky daného trojuholníka a jeho ťažiska a stredu jeho Feuerbachovej kružnice.

Výskumná úloha 34:

Zistite pomer vzdialeností ťažiska od ortocentra a od stredu opísanej kružnice.

4

Rezy kocky

4.1 Kocka

V tejto stati sa budeme zaoberať ďalšou témou stredoškolskej matematiky – rovinnými rezmi kocky. Stereometria si, samozrejme, vyžaduje ešte viac predstavivosti než planimetria, treba preto hľadať spôsoby, ako ju sprístupniť širšiemu publiku. Výbornou pomôckou tu môže byť práve program Asymptote: Ak totiž súbor s trojrozmerným obrázkom v ňom vytvoreným otvoríme v štandardnom prehliadači (Adobe Acrobat Reader DC) a tento obrázok (na výzvu) aktivujeme, budeme ho môcť (pohybom stlačenej myši) podľa potreby otáčať, a nazerať tak naň z rôznych strán. Aby sa tak mohlo udiať, v súbore `konstrukcie.tex` sa prepneme do knižnice `asydef3.tex`, ktorá obsahuje príslušné nastavenia.

- Keďže body v trojrozmernom priestore sú charakterizované troma súradnicami, budeme ich ukladať do špeciálneho typu `triple` (trojica).
- Bod (a, b, c) budeme zapisovať v tvare `(a,b,c)` (aj v tomto prípade budú hodnoty a, b a c napísané príslušným fontom).

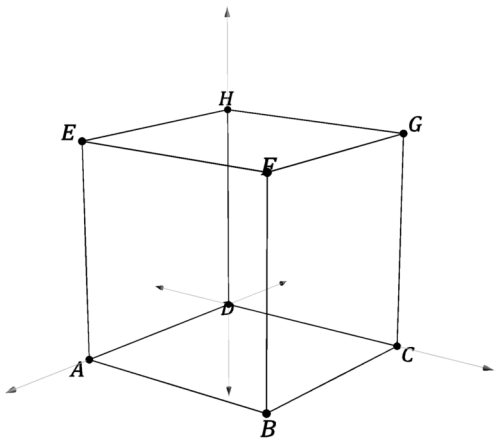
Prv než sa budeme zaoberať samotnými rezmi, potrebujeme vedieť kresliť kocku.

Príklad 28:

Nakreslite jednotkovú kocku.

Riešenie

Našu súradnicovú sústavu zvolíme tak, aby sa čo najviac podobala obvyklému pohľadu, t. j. tak, že stena $ABFE$ bude viac-menej čelná. To znamená, že jej počiatkom bude bod D a jej kladné polosi sa budú zhodovať postupne s polpriamkami DA, DC, DH.



Keďže kocka je jednotková, všetky súradnice všetkých jej vrcholov budú buď 0, alebo 1 a každá z ôsmich kombinácií týchto dvoch čísel dá iný z nich.

Po definícii polôh vrcholov už iba nakreslíme obvod každej zo šiestich stien a vrcholy označíme príslušnými písmenami.

Analógiou smerov z dvojrozmerného priestoru sú tu konštanty typu `triple` v tvaroch `d3x`, `d3y` a `d3xy`, kde x je už známe označenie niektorej hlavnej svetovej strany a y je niektoré z písmen `U` („up“ – hore) alebo `D` („down“ – dolu).

Zdôraznime, že tieto konštanty nie sú súčasťou jazyka, sú definované v knižnici `asydef3.tex`.

Aby zobrazená kocka nebola príliš malá, zmeníme aj jednotku dĺžky.

> Jednotku dĺžky možno nastaviť príkazom `unitsize` (jednotková veľkosť) nasledovaným príslušnou dĺžkou v zátvorkách.

Program na zobrazenie kocky by teda mohol vyzerať takto:

```
unitsize(3cm);

triple A = (1,0,0);
triple B = (1,1,0);
triple C = (0,1,0);
triple D = (0,0,0);
triple EE = (1,0,1);
triple F = (1,1,1);
triple G = (0,1,1);
triple H = (0,0,1);

draw(A--B--F--EE--cycle);
draw(G--H--EE--F--cycle);
draw(A--D--C--B--cycle);
draw(G--F--B--C--cycle);
draw(A--EE--H--D--cycle);
draw(G--C--D--H--cycle);

dot("$A$",A,d3SED);
dot("$B$",B,d3NED);
dot("$C$",C,d3NWD);
dot("$D$",D,d3SWD);
dot("$E$",EE,d3SEU);
dot("$F$",F,d3NEU);
dot("$G$",G,d3NWU);
dot("$H$",H,d3SWU);
```

Tieto príkazy budú pevnou súčasťou všetkých úloh tejto state. Aby sme ich nemuseli stále písať, dáme ich do knižnice `asydef3.tex` v takejto forme:

```
unitsize(3cm);

triple A = (1,0,0);
triple B = (1,1,0);
triple C = (0,1,0);
triple D = (0,0,0);
triple EE = (1,0,1);
triple F = (1,1,1);
triple G = (0,1,1);
triple H = (0,0,1);

void kresliKostruKockyABCDEFGH ()
{
  draw(A--B--F--EE--cycle);
  draw(G--H--EE--F--cycle);
  draw(A--D--C--B--cycle);
  draw(G--F--B--C--cycle);
  draw(A--EE--H--D--cycle);
  draw(G--C--D--H--cycle);
}

void kresliVrcholyKockyABCDEFGH ()
{
  dot("$A$",A,d3SED);
  dot("$B$",B,d3NED);
  dot("$C$",C,d3NWD);
  dot("$D$",D,d3SWD);
  dot("$E$",EE,d3SEU);
  dot("$F$",F,d3NEU);
  dot("$G$",G,d3NWU);
  dot("$H$",H,d3SWU);
}
```

Budeme si pritom pamätať, že premenné A, B, C, D, EE, F, G, H sú tým rezervované.

Program na zobrazenie našej kocky po tejto akcii môžeme zjednodušiť:

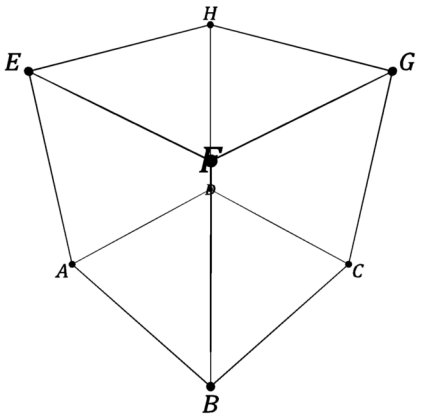
```
kresliKostruKockyABCDEFGH();

kresliVrcholyKockyABCDEFGH();
```

Uvedenú formu zobrazovania môžeme ovplyvniť takto:

- Spôsob projekcie je uložený v systémovej premennej `currentprojection` (aktuálne premietanie).
- Perspektívne zobrazovanie s polohou pozorovateľa (x, y, z) je hodnotou príkazu `perspective` (perspektíva) s troma vstupmi x, y, z.
- Rovnobežné premietanie s polohou pozorovateľa (x, y, z) je hodnotou príkazu `orthographic` (rovnobežné) s troma vstupmi x, y, z.

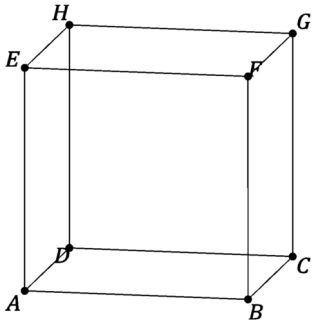
Ak z nejakých dôvodov chceme, aby bol vrchol D našej kocky zobrazovaný vľavo dolu, náš program môžeme upraviť napríklad takto:

```
currentprojection = perspective(1.5,1.5,1.5);

kresliKostruKockyABCDEFGH();

kresliVrcholyKockyABCDEFGH();
```



V pravouhlom zobrazení to bude vyzerať takto:

```
currentprojection = orthographic(2.5,0.5,0.5);

kresliKostruKockyABCDEFGH();

kresliVrcholyKockyABCDEFGH();
```



My však z estetických i praktických dôvodov ostaneme pri pôvodnom perspektívnom zobrazení a v prednastavenej polohe.

4.2 Rezy

Ešte predtým, než sa sústredíme na samotné rezy, si pripomeňme niekoľko základných vedomostí:

- Pod *rezom* telesa rovinou rozumieme ich prienik.
- Útvar nazývame *konvexný*, ak s každými dvoma bodmi obsahuje aj úsečku, ktorá je nimi určená.

  Uvedomme si, že všetky útvary, s ktorými budeme pracovať – rovina (špeciálne rovina rezu), kocka, jej steny i jej hrany – sú konvexné.
- Prienik konvexných útvarov je tiež konvexný – s každými dvoma ich spoločnými bodmi totiž do každého z nich patrí i úsečka týmito bodmi určená.

  To znamená, že každý rez kocky je konvexný útvar.
- V rovine nazývame *hraničným* taký bod útvaru tejto roviny, že každý kruh so stredom v ňom obsahuje aj nejaký bod tohto útvaru aj nejaký bod mimo neho. Množina hraničných bodov útvaru sa nazýva jeho *hranicou*.

  Napríklad hranica mnohouholníka je tvorená práve jeho stranami, hranica kruhu je jeho obvodová kružnica, hranicou polroviny je jej hraničná priamka, kým celá rovina žiaden hraničný bod nemá.
- Analogické definície platia i v (trojrozmernom) priestore, len v prvej treba kruh nahradiť guľou.

  Hranica kocky je teda tvorená práve jej stenami.
- Prienik dvoch rôznobežných rovín je priamka, ktorú nazývame *priesečnica*.
- Rezom konvexného mnohostenu (špeciálne kocky) je konvexný mnohouholník.

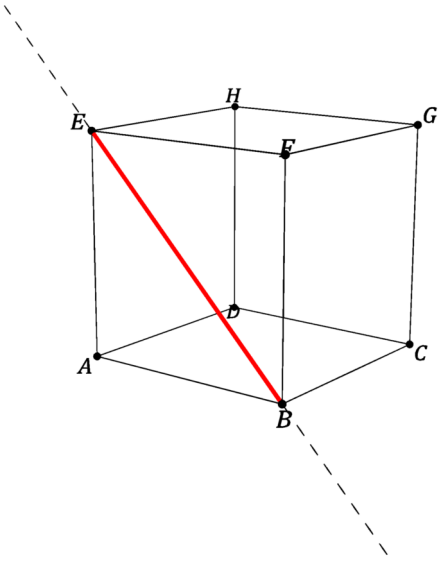
Tieto poznatky nám napovedajú, ako hľadať rez kocky danou rovinou: treba nájsť jeho hranicu, ktorá je tvorená (neprázdnymi) prienikmi roviny rezu s jednotlivými stenami kocky.

Príklad 29:

Zostrojte rez jednotkovej kocky $ABCDEFGH$ rovinou EBG.

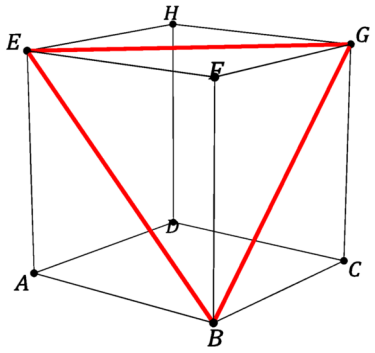
Riešenie

Všimnime si, že rovina rezu a rovina steny $EABF$ sú rôznobežné – body E a B ležia v oboch, kým bod G len v prvej z nich. Priesečnica týchto rovín je teda priamka EB, a keďže body E a B ležia na hranici kocky, prienikom priesečnice (a teda roviny rezu) so stenou $EABF$ je práve úsečka EB. Tá je preto časťou hranice nášho rezu.



Analogickou úvahou pre steny $BCGF$ a $GHEF$ dostávame ďalšie dve strany rezu, a to BG a GE. Keďže krivka $EBGE$ je uzavretá, sú to všetky strany rezu. Hľadaným rezom je teda trojuholník EBG.

```
kresliKostruKockyABCDEFGH();

draw(EE--B--G--cycle,red+linewidth(1.5));

kresliVrcholyKockyABCDEFGH();
```

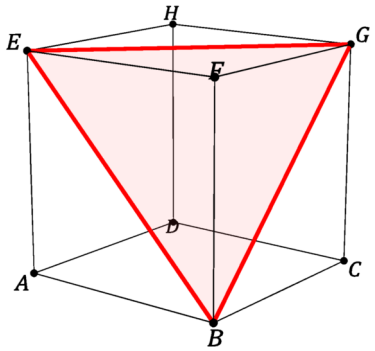


Aby bol tento rez lepšie viditeľný, vyfarbíme ho. Tu, žiaľ, nepomôže príkaz `fill`, známy z dvojrozmerných obrázkov. Najprv musíme definovať vyfarbovanú plochu, a to pomocou krivky, ktorá je jej hranicou.

- Krivky v trojrozmernom priestore majú dátový typ `path3` (cesta v 3D).
- Plochy majú dátový typ `surface` (plocha).
- Na vytvorenie istej plochy ohraničenej danou uzavretou krivkou (samozrejme, podľa istých pravidiel, veď takýchto plôch existuje nekonečne veľa) sa používa (rovnomenný) príkaz `surface` (plocha), ktorého vstupom je táto krivka. V našom prípade to bude vždy rez čiže podmnožina roviny, spomínané pravidlá preto zaručia, že aj takto vzniknutá plocha bude ležať v tej istej rovine.

Vytvoríme teda hraničnú krivku `obvodPlochy` a pomocou nej plochu `plocha`. Obe vyfarbíme:

```
path3 obvodPlochy = EE--B--G--cycle;

surface plocha = surface(obvodPlochy);

kresliKostruKockyABCDEFGH();

draw(plocha,red+opacity(.1));
draw(obvodPlochy,red+linewidth(1.5));

kresliVrcholyKockyABCDEFGH();
```



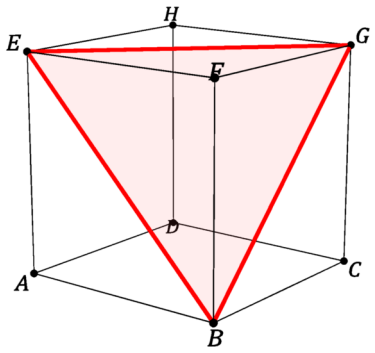
Všimnime si, že zo štvorice pridaných riadkov informáciu nesie akurát prvý, kde je uvedený mnohouholník rezu,

i keď možno môžeme ovplyvniť i farbu rezu či jeho obvodu. Vytvoríme si preto takýto príkaz, ktorý hneď vložíme do zásobárne `asydef3.tex`:

```
void kresliPlochu (path3 obvod,
  pen farba=red,
  real nepriehladnost=.1,
  real hrubka=1.5)
{
  surface plocha = surface(obvod);
  draw(plocha,farba+opacity(nepriehladnost));
  draw(obvod,farba+linewidth(hrubka));
}
```

Za zmienku tu stoja tri parametre typu `pen` (pero) i to, že majú preddefinovanú hodnotu. A tak napríklad volanie `kresliPlochu(EE--B--G--cycle,yellow,blue,linewidth(2))` vykreslí rez nepriehľadnou žltou farbou s hrubým modrým obvodom, kým pri jednoduchom volaní `kresliPlochu(EE--B--G--cycle)` sa použijú preddefinované hodnoty, takže rez bude priehľadne červený a bude mať sýtočervený obvod primeranej hrúbky.

```
kresliKostruKockyABCDEFGH();

kresliPlochu(EE--B--G--cycle);

kresliVrcholyKockyABCDEFGH();
```



Výskumná úloha 35:

Aká je vzájomná poloha rovín ACH a BEG v kocke $ABCDEFGH$?

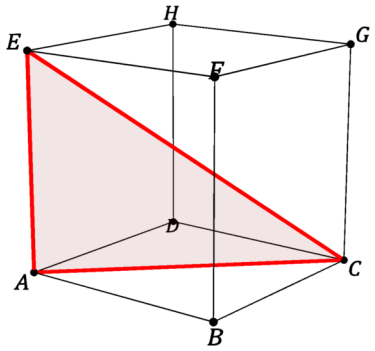
Príklad 30:

Zostrojte rez jednotkovej kocky $ABCDEFGH$ rovinou ACE.

Riešenie

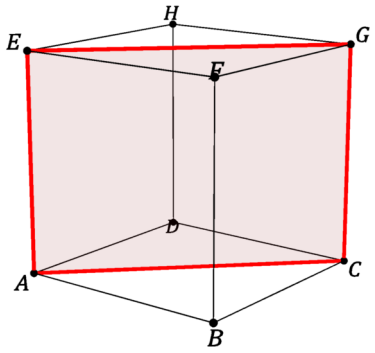
Z riešenia predchádzajúcich úloh by azda mohol niekto nadobudnúť dojem, že rezom kocky rovinou danou troma bodmi je vždy trojuholník s týmito troma vrcholmi. Ako však uvidíme na tomto príklade, nemusí to tak byť. Jedna zo strán trojuholníka ACE – úsečka EC – je totiž telesovou uhlopriečkou našej kocky, a teda leží (až na krajné body) v jej vnútri:

```
kresliKostruKockyABCDEFGH();

kresliPlochu(A--C--EE--cycle);

kresliVrcholyKockyABCDEFGH();
```



Rovina rezu obsahuje body A a E, je teda rôznobežná s oboma rovinami obsahujúcimi dolnú stenu $ABCD$ i hornú stenu $EFGH$. Tie sú rovnobežné, aj jej priesečnice s nimi teda musia byť rovnobežné. V rovine $ABCD$ je to priamka AC, takže v rovine to musí byť priamka EG. Obvod rezu teda obsahuje úsečky AC a EG, a keďže AE a CG sú zároveň hranami kocky, aj ony sú jeho súčasťou. Zhrnutím dostávame, že obvod rezu je štvoruholník (presnejšie obdĺžnik) $ACGE$.

```
kresliKostruKockyABCDEFGH();

kresliPlochu(A--C--G--EE--cycle);

kresliVrcholyKockyABCDEFGH();
```



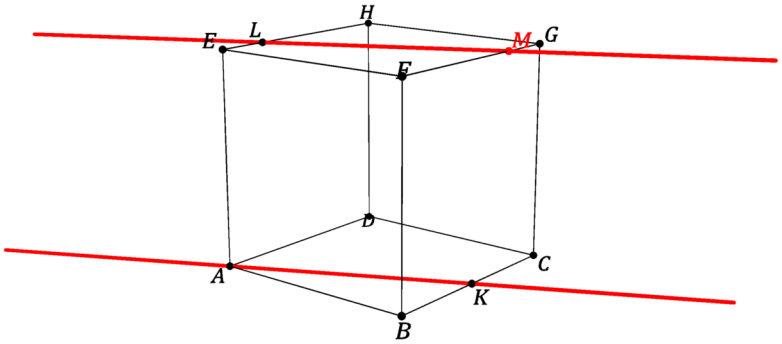
Rovina rezu nemusí byť zadaná len vybranými vrcholmi kocky, napríklad v nasledujúcom príklade sú body rezu určené polohou na niektorých hranách.

Príklad 31:

Zostrojte rez jednotkovej kocky $ABCDEFGH$ rovinou AKL, kde K je stred hrany BC a L je bod hrany EH taký, že $|EL| = \frac{1}{4}$.

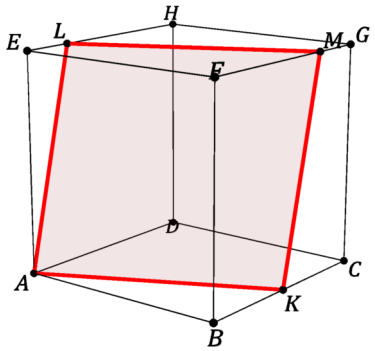
Riešenie

Rovina rezu pretína rovnobežné steny $ABCD$ a $EFGH$ v dvoch rovnobežných priamkach. Prvá je AK, druhá prechádza bodom L. Vzhľadom na polohu bodu L na úsečku HE táto priamka pretína aj úsečku FG. Ich priesečník označme M:



Keďže úsečky AK a LM ležia ako v rovine rezu, tak na povrchu kocky, sú súčasťou obvodu rezu. To isté platí pre úsečky AL a KM, takže hľadaným rezom je štvoruholník $AKML$. Vzhľadom na rovnobežnosť príslušných stien kocky je to navyše rovnobežník, preto platí $\overrightarrow{LM} = \overrightarrow{AK}$, čo umožňuje jednoduchšiu konštrukciu bodu M bez pomocných priamok:

```
triple K = midpoint(B--C);
triple L = point(EE--H,1/4);

triple M = L+(K-A);

kresliKostruKockyABCDEFGH();

kresliPlochu(A--K--M--L--cycle);

kresliVrcholyKockyABCDEFGH();

dot("$K$",K,d3ND);
dot("$L$",L,d3SU);
dot("$M$",M,d3NU);
```



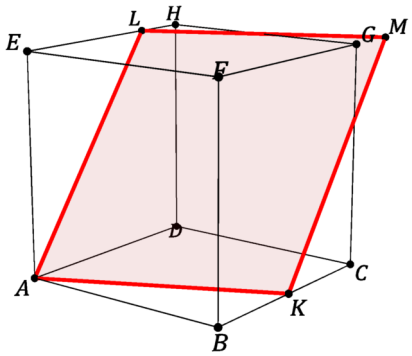
Príklad 32:

Zostrojte rez jednotkovej kocky $ABCDEFGH$ rovinou AKL, kde K je stred hrany BC a L je bod hrany HE taký, že $|EL| = \frac{3}{4}$.

Riešenie

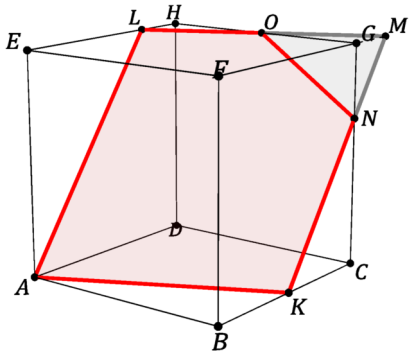
Odrazme sa od riešenia predchádzajúceho príkladu, samozrejme po zmene `1/4` na `3/4`:

```
triple K = midpoint(B--C);
triple L = point(EE--H,3/4);

triple M = L+(K-A);

kresliKostruKockyABCDEFGH();

kresliPlochu(A--K--M--L--cycle);

kresliVrcholyKockyABCDEFGH();

dot("$K$",K,d3ND);
dot("$L$",L,d3SU);
dot("$M$",M,d3NU);
```



Ako vidieť, vrchol M rovnobežníka $BKML$ tentoraz neleží na úsečke FG, ale na jej predĺžení. To však znamená, že úsečka KM pretne úsečku CG a analogicky úsečka LM pretne úsečku HG, a to v bodoch N, resp. O. A keďže každá z dvojice bodov K s N, N s O a O s L leží v niektorej zo stien kocky, rezom bude päťuholník $AKNOL$:

```
triple K = midpoint(B--C);
triple L = point(EE--H,3/4);

triple M = L+(K-A);

triple NN = intersectionpoints(K--M,C--G)[0];
triple O = intersectionpoints(L--M,H--G)[0];

kresliKostruKockyABCDEFGH();

draw(M--G,dashed);

kresliPlochu(A--K--NN--O--L--cycle);
kresliPlochu(M--NN--O--cycle,gray);

kresliVrcholyKockyABCDEFGH();

dot("$K$",K,d3ND);
dot("$L$",L,d3SU);
dot("$M$",M,d3NU);
dot("$N$",NN,d3NW);
dot("$O$",O,d3NU);
```



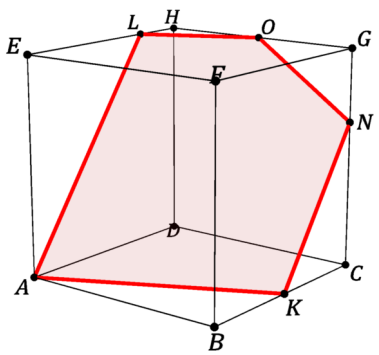
Aby sme vyhli komplikovaným konštrukciám dvojíc rovnobežných priamok, použijeme nasledujúcu analógiu príkazu `extension` z dvojrozmerného prípadu:

Jediný priesečník dvoch rôznobežných priamok v (trojrozmernom) priestore možno získať ako výsledok príkazu `priesecnikRoznobeziek`, ktorého prvé dva argumenty sú dva rôzne body prvej priamky a druhé dva rôzne body druhej.

Zdôraznime, že tento príkaz (ako to naznačuje už jeho slovenský názov) nie je súčasťou jazyka Asymptote, je to iba náš doplnok v knižnici `asydef3.tex`.

Použitím tohto príkazu môžeme náš program zjednodušiť, pričom nenakreslíme už ani pomocný bod M:

```
triple K = midpoint(B--C);
triple L = point(EE--H,3/4);

triple M = L+(K-A);

triple NN = priesecnikRoznobeziek(K,M, C,G);
triple O = priesecnikRoznobeziek(L,M, H,G);

kresliKostruKockyABCDEFGH();

kresliPlochu(A--K--NN--O--L--cycle);

kresliVrcholyKockyABCDEFGH();

dot("$K$",K,d3ND);
dot("$L$",L,d3SU);
dot("$N$",NN,d3NW);
dot("$O$",O,d3NU);
```



Výskumná úloha 36:

Nech $ABCDEFGH$ je jednotková kocka. Nech $0 \leq k, l, m \leq 1$ a $m \leq k, l$. Nech K leží na hrane BC, L na EH a M na AD, pričom platí $|BK| = k$, $|EL| = l$, $|AM| = m$. Zistite, akú podmienku spĺňajú parametre k, l, m, keď je rez kocky $ABCDEFGH$ rovinou KLM

a) štvoruholník;

b) päťuholník.

Výskumná úloha 37:

Nech $KLNM$ je rez jednotkovej kocky $ABCDEFGH$ rovinou KLM, kde K je zvolený bod hrany DA, L je zvolený bod hrany DC a M je zvolený bod hrany HE taký, že $|HM| \leq |DK|$. Zistite, aká je vzájomná poloha priamok DH, KM a LN, ak platí:

a) $|HM| = |DK|$;

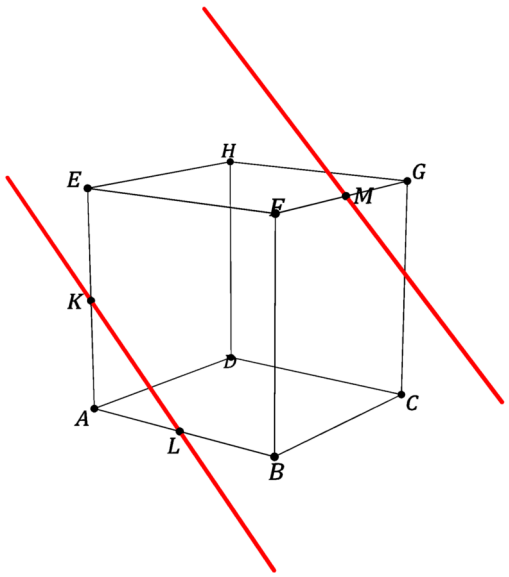
b) $|HM| < |DK|$.

Príklad 33:

Zostrojte rez jednotkovej kocky $ABCDEFGH$ rovinou KLM, kde K je stred AE, L je stred AB a M je stred FG.

Riešenie

Sme teraz v novej situácii, lebo priamka rovnobežná s KL a prechádzajúca bodom M má s kockou jediný spoločný bod (konkrétne M):

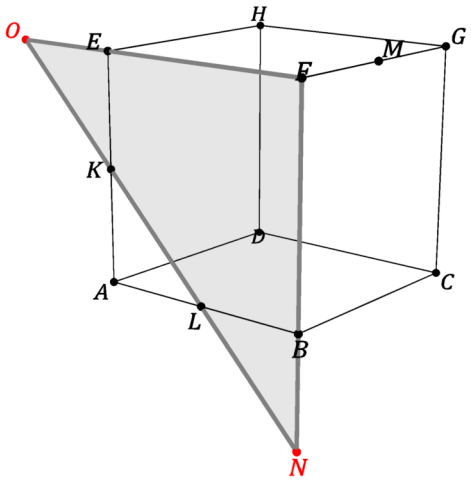


Musíme na to teda ísť inak.

Keďže K patrí do roviny steny $DAEH$, ale L do nej nepatrí, priamka KL je rôznobežná s touto rovinou, a teda aj s rovinou steny $BCGF$, ktorá je s ňou rovnobežná. Znamená to, že táto priamka a táto rovina majú spoločný bod, ktorý označme N. Tento bod leží na priamke KL, a teda i v rovine steny $ABFE$, v ktorej táto priamka leží. Bod N teda leží v oboch rovinách – ako v rovine steny $ABFE$, tak v rovine steny $BCGF$ –, takže leží na ich priesečnici FB. To dáva návod na konštrukciu bodu N: je to priesečník priamok KL a FB.

Analogicky zostrojíme bod O ako priesečník priamok KL a FE:

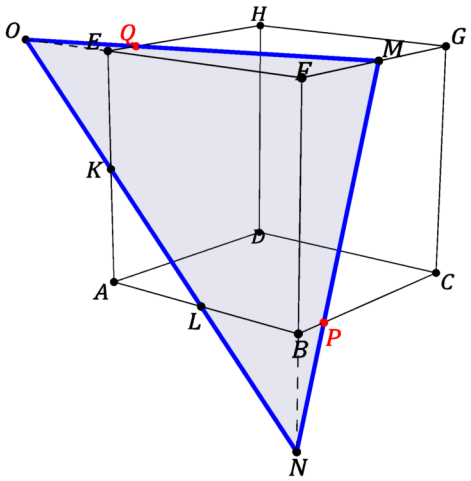
```
triple K = midpoint(A--EE);
triple L = midpoint(A--B);
triple M = midpoint(F--G);

triple NN = priesecnikRoznobeziek(K,L, F,B);
triple O = priesecnikRoznobeziek(K,L, F,EE);

kresliKostruKockyABCDEFGH();

kresliPlochu(F--NN--O--cycle,gray);

kresliVrcholyKockyABCDEFGH();

dot("$K$",K,d3SE);
dot("$L$",L,d3ED);
dot("$M$",M,d3NU);

dot("$N$",NN,d3NED,red);
dot("$O$",O,d3SEU,red);
```



Body M a N z roviny rezu ležia aj v rovine steny $BCGF$, takže v nej leží celá priamka MN. Jednou z hrán rezu teda bude jej prienik MP so stenou $BCGF$, kde P je priesečník priamky MN a úsečky BC.

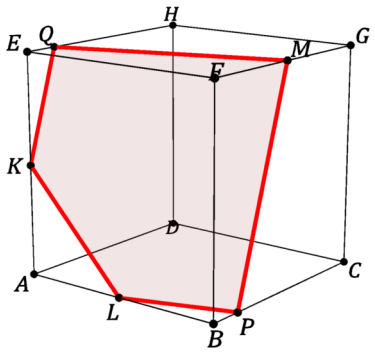
Analogicky bude ďalšou hranou rezu MQ, kde Q je priesečník priamky MO a úsečky EH:

```
triple K = midpoint(A--EE);
triple L = midpoint(A--B);
triple M = midpoint(F--G);

triple NN = priesecnikRoznobeziek(K,L, F,B);
triple O = priesecnikRoznobeziek(K,L, F,EE);

triple P = priesecnikRoznobeziek(M,NN, B,C);
triple Q = priesecnikRoznobeziek(M,O, EE,H);

kresliKostruKockyABCDEFGH();

draw(EE--O,dashed);
draw(B--NN,dashed);

kresliPlochu(M--NN--O--cycle,blue);

kresliVrcholyKockyABCDEFGH();

dot("$K$",K,d3SE);
dot("$L$",L,d3ED);
dot("$M$",M,d3NU);

dot("$N$",NN,d3NED);
dot("$O$",O,d3SEU);

dot("$P$",P,d3ND,red);
dot("$Q$",Q,d3EU,red);
```



A keďže úsečky KL, LP a KQ ležia v stenách kocky, hľadaným rezom je päťuholník $KLPMQ$ (pomocné body N a O už nevykresľujeme):

```
triple K = midpoint(A--EE);
triple L = midpoint(A--B);
triple M = midpoint(F--G);

triple NN = priesecnikRoznobeziek(K,L, F,B);
triple O = priesecnikRoznobeziek(K,L, F,EE);

triple P = priesecnikRoznobeziek(M,NN, B,C);
triple Q = priesecnikRoznobeziek(M,O, EE,H);

kresliKostruKockyABCDEFGH();

kresliPlochu(K--L--P--M--Q--cycle);

kresliVrcholyKockyABCDEFGH();

dot("$K$",K,d3SE);
dot("$L$",L,d3ED);
dot("$M$",M,d3NU);

dot("$P$",P,d3ND);
dot("$Q$",Q,d3EU);
```



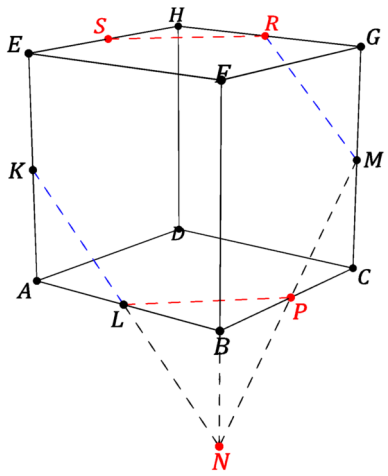
Príklad 34:

Zostrojte rez jednotkovej kocky $ABCDEFGH$ rovinou KLM, kde K je stred AE, L je stred AB a M je stred CG.

Riešenie

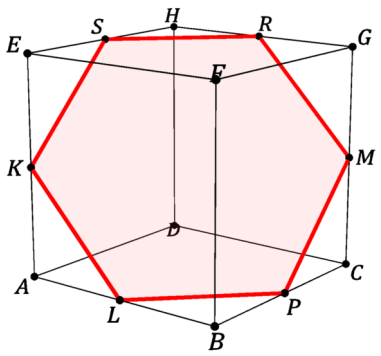
Postupujeme podobne ako v predchádzajúcej úlohe, rovnakým spôsobom zostrojíme bod N ako priesečník priamok KL a FB a bod P ako priesečník priamok MN a BC. Tu sa však podobnosť končí, lebo bod M neleží na stene $EFGH$, takže postup konštrukcie bodu O (a tým pádom ani Q) už nemôžeme zopakovať. Na tejto stene, presnejšie na jej hrane GH, však môžeme zostrojiť bod R, a to pomocou rovnobežky s KL cez bod M. Podobne získame posledný vrchol rezu – S, ten bude analogicky ležať na rovnobežke s PL cez R:

```
triple K = midpoint(A--EE);
triple L = midpoint(A--B);
triple M = midpoint(C--G);

triple NN = priesecnikRoznobeziek(K,L, F,B);
triple P = priesecnikRoznobeziek(M,NN, B,C);
triple R = priesecnikRoznobeziek(M,M+(L-K), H,G);
triple SS = priesecnikRoznobeziek(R,R+(L-P), H,EE);

kresliKostruKockyABCDEFGH();

draw(L--NN,dashed);
draw(M--NN,dashed);
draw(B--NN,dashed);
draw(K--L,blue+dashed);
draw(M--R,blue+dashed);
draw(P--L,red+dashed);
draw(R--SS,red+dashed);

kresliVrcholyKockyABCDEFGH();

dot("$K$",K,d3SE);
dot("$L$",L,d3ED);
dot("$M$",M,d3NW);
dot("$N$",NN,d3NED,red);
dot("$P$",P,d3ND,red);
dot("$R$",R,d3WU,red);
dot("$S$",SS,d3SU,red);
```



Rezom je teda šesťuholník $KLPMRS$:

```
triple K = midpoint(A--EE);
triple L = midpoint(A--B);
triple M = midpoint(C--G);

triple NN = priesecnikRoznobeziek(K,L, F,B);
triple P = priesecnikRoznobeziek(M,NN, B,C);
triple R = priesecnikRoznobeziek(M,M+(L-K), H,G);
triple SS = priesecnikRoznobeziek(R,R+(L-P), H,EE);

kresliKostruKockyABCDEFGH();

kresliPlochu(K--L--P--M--R--SS--cycle);

kresliVrcholyKockyABCDEFGH();

dot("$K$",K,d3SE);
dot("$L$",L,d3ED);
dot("$M$",M,d3NW);
dot("$P$",P,d3ND);
dot("$R$",R,d3WU);
dot("$S$",SS,d3SU);
```



Úloha 13:

Zostrojte rez jednotkovej kocky $ABCDEFGH$ rovinou KLM, kde K je stred AE, L je stred AB a M je stred BC.

Príklad 35:

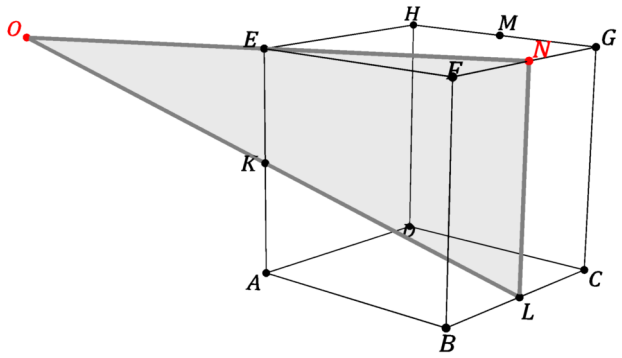
Zostrojte rez jednotkovej kocky $ABCDEFGH$ rovinou KLM, kde K je stred AE, L je stred CB a M je stred HG.

Riešenie

Teraz sme v úplne novej situácii, lebo žiadna z úsečiek KL, LM a MK neleží v žiadnej stene, priame využitie rovnobežnosti priesečníc preto zatiaľ neprichádza do úvahy. Musíme si poradiť inak:

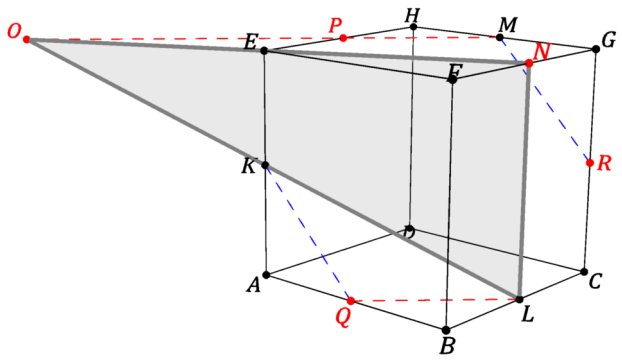
Keďže priamka KE je rôznobežná s rovinou $EFGH$ (obsahuje totiž jej bod E, ale aj bod K, ktorý v nej neleží), s ňou rovnobežná priamka prechádzajúca bodom L musí byť s touto rovinou rôznobežná tiež. Ich priesečník označme N. Rovnobežky KE a LN potom ležia v tej istej rovine KLE, ležia tam preto i priamky LK a NE. Tie sú rôznobežné, lebo rovina steny $EFGH$ obsahuje NE, ale s ňou rovnobežná rovina steny $ABCD$ obsahuje L, nie však K. Ich priesečník – označme ho O – leží ako v rovine rezu KLM (lebo patrí jej priamke KL), tak v rovine steny $EFGH$ (lebo patrí jej priamke NE).

```
triple K = midpoint(A--EE);
triple L = midpoint(C--B);
triple M = midpoint(H--G);

triple NN = L+(EE-A);
triple O = priesecnikRoznobeziek(L,K, NN,EE);

kresliKostruKockyABCDEFGH();

kresliPlochu(L--NN--O--cycle,gray);

kresliVrcholyKockyABCDEFGH();

dot("$K$",K,d3SE);
dot("$L$",L,d3ND);
dot("$M$",M,d3WU);

dot("$N$",NN,d3NU,red);
dot("$O$",O,d3SEU,red);
```



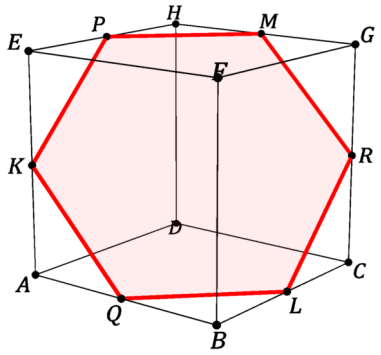
Teraz už môžeme pokračovať obvyklým spôsobom: Ďalším vrcholom (P) rezu je priesečník úsečiek MO a HE a zvyšné dva (Q a R) získame pomocou rovnobežiek.

```
triple K = midpoint(A--EE);
triple L = midpoint(C--B);
triple M = midpoint(H--G);

triple NN = L+(EE-A);
triple O = priesecnikRoznobeziek(L,K, NN,EE);
triple P = priesecnikRoznobeziek(EE,H, O,M);
triple Q = priesecnikRoznobeziek(A,B, L,L+(P-M));
triple R = priesecnikRoznobeziek(C,G, M,M+(Q-K));

kresliKostruKockyABCDEFGH();

kresliPlochu(L--NN--O--cycle,gray);

draw(O--M,red+dashed);
draw(Q--L,red+dashed);
draw(K--Q,blue+dashed);
draw(M--R,blue+dashed);

kresliVrcholyKockyABCDEFGH();

dot("$K$",K,d3SE);
dot("$L$",L,d3ND);
dot("$M$",M,d3WU);
dot("$N$",NN,d3NU,red);
dot("$O$",O,d3SEU,red);
dot("$P$",P,d3SU,red);
dot("$Q$",Q,d3ED,red);
dot("$R$",R,d3NW,red);
```



Hľadaným rezom je teda šesťuholník $KQLRMP$ (pomocné úsečky a body N a O už nevykresľujeme):

```
triple K = midpoint(A--EE);
triple L = midpoint(C--B);
triple M = midpoint(H--G);

triple NN = L+(EE-A);
triple O = priesecnikRoznobeziek(L,K, NN,EE);
triple P = priesecnikRoznobeziek(EE,H, O,M);
triple Q = priesecnikRoznobeziek(A,B, L,L+(P-M));
triple R = priesecnikRoznobeziek(C,G, M,M+(Q-K));

kresliKostruKockyABCDEFGH();

kresliPlochu(K--Q--L--R--M--P--cycle);

kresliVrcholyKockyABCDEFGH();

dot("$K$",K,d3SE);
dot("$L$",L,d3ND);
dot("$M$",M,d3WU);
dot("$P$",P,d3SU);
dot("$Q$",Q,d3ED);
dot("$R$",R,d3NW);
```



Register