Obsah

Doporučujeme si vytisknout popis, který je uložen v adresáři Popis. Obsah tohoto dokumentu je stejný jako u této nápovědy, ale tento dokument je upraven pro tisk.

Kontakt na autora:
Petr Plavjaník
Heyrovského 974
67405 Třebíč
plavjanik@email.cz
http://alias.trb.czn.cz

Úvod

Do deskriptivní geometrie patří dva velké obory zobrazení prostorových útvarů do roviny a řešení stereometrických problémů.
Pro zobrazení trojrozměrných těles se ve středoškolské geometrii používá *rovnoběžné promítání.* Promítají se tyto geometrické útvary: bod, přímka, rovina, mnohoúhelníky, mnohostěny, kuželosečky, jednoduchá oblá tělesa, koule, kužel a válec. Mezi *stereometrické problémy* patří například určení viditelnosti těchto těles, průniky těles s přímkou, řez rovinou a nejvyšším stupněm je průnik i několika prostorových těles.
Při řešení těchto úloh se provádí jednotlivé kroky – *konstrukce* – pomocí pravítka a kružítka. Jejich pořadí je takové, abychom dospěli k cíli. Mezi základní konstrukce, ze kterých se skládá každý postup, patří podle učebnice [1]:
konstrukce roviny, která je určena a) třemi body neležícími v přímce; b) přímkou a bodem neležícím na této přímce; c) dvěma různoběžkami; d) dvěma nesplývajícími rovnoběžkami.
planimetrické konstrukce v každé rovině.

- konstrukce průsečnice dvou různých rovin (pokud existuje).
- konstrukce průsečíku přímky s rovinou (pokud existuje).
- konstrukce roviny vedené daným bodem kolmo k dané přímce
- konstrukce přímky vedené daným bodem kolmo k dané rovině.
- konstrukce přímky vedené daným bodem rovnoběžně s danou přímkou.
- konstrukce roviny vedené daným bodem rovnoběžně s danou rovinou.

U jednodušších úloh (jejichž řešení obsahuje jen několik konstrukcí) je postup zřejmý a většinou jej nezapisujeme. Ale u úloh obtížnějších je třeba si postup promyslet a poznamenat ještě před vlastním rýsováním. Protože rýsování je časově náročné, kreslíme si při vymýšlení postupu náčrtek, do kterého zakreslujme jednotlivé kroky.
Pokud pro danou úlohu existuje postup neboli *algoritmus*, je možné takový postup zapsat do počítače, který by ho vyrýsoval. Tím by odpadla nejzdlouhavější práce a mohli bychom se soustředit jen na řešení úlohy. Aby počítač mohl postup pochopit, musí býti stanovena pravidla, podle kterých bude možno postup jednoznačně zapisovat. Samozřejmě musí existovat program, který takto zapsaný postup promění v žádaný výkres. Takovým software je předložený program *Deskriptivní geometrie*.

Účel programu

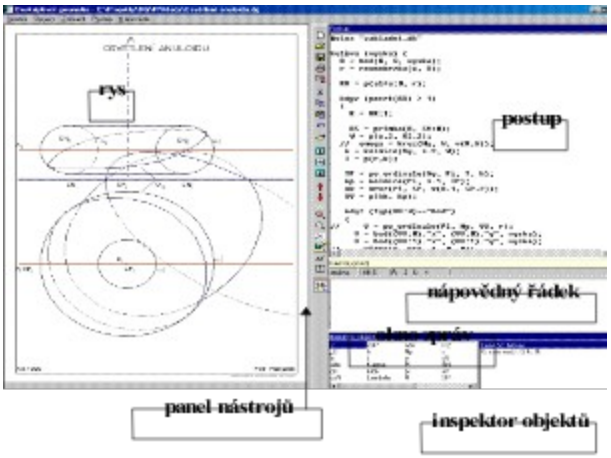
Program Deskriptivní geometrie byl vytvořen k usnadnění řešení úloh z deskriptivní geometrie. Základní princip spočívá ve vyrýsování podle postupu, který je zapsán uživatelem. Postup je zapisován v programovacím jazyce odvozeného z jazyka C, pozměněného tak, aby byl více vhodný pro řešení deskriptivních úloh.
Oproti tradičnímu rýsování na papíře má použití programu mnoho výhod:

- přesnost, všechny výpočty založeny na analytické geometrie a počítány s velice vysokou přesností, které je schopný dosáhnout jen počítač
- rychlost, rýsování programu je téměř okamžité
- flexibilita, cokoliv lze změnit, změna jednoho bodu způsobí změnu všeho, co se k němu vztahuje
- zápis složitých postupů (algoritmů), programovací jazyk umožňuje zápis strukturovaných příkazů (podmínek, cyklů) a funkcí
- jednoduchá orientace ve složitých rysech, najít tu správnou čáru je snadné díky vysvícení těch objektů, se kterými se právě pracuje

Technické informace o programu

Program byl vytvořen ve vývojovém prostředí *Borland Delphi 3 pro Windows 95*, které umožňuje vytváření aplikací pro Windows 95. Tudíž i program *Deskriptivní geometrie* požaduje ke svému běhu operační systém *Windows 95*. Tento systém přináší s sebou výhody a nevýhody. Mezi nejdůležitější výhody patří nezávislost na použitém hardware (je jedno jakou grafickou kartu, monitor a hlavně tiskárnu používáte) a snadné ovládání, které je podobné ostatním aplikacím v tomto systému. Nevýhodou pak je větší velikost programu.
Požadavky na vybavení počítače se odvíjejí od požadavků operačního systému. Aby bylo na programu možné pohodlně pracovat je zapotřebí počítače s procesorem *Pentium* a operační pamětí nejméně 16 MB. Nároky na pevný disk jsou malé (okolo dvou megabajtů). Důležitá je i velikost monitoru. Protože se jedná o grafický program, je pro dosažení kvalitního zobrazení zapotřebí většího monitoru (nejméně 15”) s vyšším rozlišením (vhodným rozlišením je nejméně 1024x768, menší rozlišení se projeví na kvalitě).
Nároky na hardware nejsou přehnané a na dnes vyráběných počítačích (Pentium MMX, 32 MB RAM, 15” monitor, rychlá 2D (3D) grafická karta) běží program svižně a kvalita zobrazení je na dobré úrovni.

Popis prostředí programu Deskriptivní geometrie



Hlavní okno Deskriptivní geometrie obsahuje jako každá Windows aplikace nabídkový pruh – *menu* –, dále obsahuje *panely nástrojů* (tlačítkové lišty), které lze libovolně přesouvat nebo je schovat.
Vlevo je zobrazení *rysu* a vpravo textové pole pro zápis *postupu*. Pod postupem je ještě okno *zpráv* od překladače a programu; úplně dole je *inspektor objektů*. Poměry velikostí jednotlivých částí lze libovolně měnit tak, aby vyhovovaly uživateli.

Rys
Rys je výsledkem činnosti programu, je jím myšlen papír o velikostech nastavitelných z programu (více viz kapitola *Nastavení rysu*). Rys lze zvětšovat a zmenšovat, případně jím posouvat.
Zvětšení či zmenšení lze provést přes nabídku Rys, nebo pomocí panelu nástrojů Rys.
Při pohybu ukazatele myší nad rysem dochází k zobrazení souřadnic pod ukazatelem v nápovědním řádku. Pokud je v *inspektoru objektů* vybrán nějaký bod, jsou zobrazeny i souřadnice relativně k tomuto bodu. Při kliknutí levým tlačítkem dojde k vygenerování kódu, který vytvoří bod o těchto souřadnicích v půdorysu (`bod(x, y, 0)`). Pravé tlačítko vytvoří nárys. Držíte-li při kliknutí klávesu Ctrl, bude vytvořen bod s relativními souřadnicemi (`A.1 + bod(x, y, 0)`).

Postup
Jedná se o textové pole, které může obsahovat i formátovaný text (*Rich Text Format*). Jeho ovládání je zcela standardní. Text lze označit do bloku, kopírovat, vkládat atd. Tyto příkazy jsou přístupné přes menu Úpravy nebo panel nástrojů a hlavně pomocí zkratkových kláves (Ctrl+X, C, V).
Mezi další funkce patří vyhledání textu a jeho případné nahrazení textem jiným.
Pod tímto textovým polem se nachází tzv. *nápovědný řádek* světle žluté barvy, jehož obsah se mění podle pozice kurzoru v textu. Pokud je nad nějakou funkcí, zobrazí se v tomto řádku její stručný popis a parametry.
Jestliže se nachází nad názvem nějaké skupiny, je tato skupina označena v inspektoru objektů a zvýrazněna na rysu.
Je vhodné před začátkem psaní postupu, přepnout si klávesnici na americkou, protože na ní se lépe píší číslice a symboly.

Pokud vyvoláte *popup menu* (stiskem prvého tlačítka myši nebo speciální klávesy na klávesnici), objeví se menu se seznamem všech funkcí, uživatelské funkce jsou na začátku seznamu a jsou označeny `*`. Po vybrání některé položky, bude název této funkce napsán do postupu.

Zprávy
Tento seznam obsahuje zprávy od překladače (chybová hlášení, pokud nastala chyba, jinak obsahuje dobu, po kterou překlad probíhal), nebo zprávu od programu (viz příkaz `text`). Každý řádek začíná číslem řádku programu, ke kterému zpráva náleží. Po vybrání určitého řádku ze seznamu se tento řádek v postupu zvýrazní.

Inspektor objektů
Je seznamem všech skupin objektů, které jsou seřazeny buď abecedně, nebo tak, jak vznikly. Po vybrání jakékoliv skupiny se objeví v pravé části její popis (blok, ve kterém vznikla) a seznam objektů, které do ní patří. U každého objektu je jeho číslo (stejné číslo, které se používá v programovacím jazyce), typ a číslo řádku. U některých objektů je napsána i jejich hodnota (u čísel, textu, bodů a vektorů). Při vybrání skupiny se zvýrazní tato skupina i na rysu. Pokud chcete zvýraznit jen jeden objekt, vyberte jej v seznamu. Stiskem pravého tlačítka se ukáže ta část kódu, ve kterém označený objekt vznikl (tento řádek bude třetí seshora). Poklepáním na nějaký objekt se vloží do kódu název jeho skupiny a případně i jeho číslo.

Nastavení
Dialogové okno *Nastavení programu* se skládá z několika stránek, v nichž můžete nastavit vlastnosti programu týkající se těchto oblastí:
Editor
Lze nastavit font, který bude použit v editoru, jeho velikost a barva. Protože v postupu můžete použít i více fontů zároveň (jiný pro řádky, které něco dělají, jiný pro komentáře), je třeba text postupu přeformátovat příkazem *Přeformátování textu* v menu *Úpravy*. Tento příkaz nastaví zvolený font u všech řádků kromě komentářů.
Dále lze nastavit automatické odsazování, který začne nový řádek na tom sloupci, kde začínal předcházející řádek. Navíc pokud končí řádek znakem `{`, bude nový řádek odsazen o dvě mezery dál.
Pomocí klávesy F3 nebo příkazu *Malý rys/Velký rys* lze rychle měnit poměr velikosti rysu ku editoru. Na této stránce můžete tyto poměry nastavit v procentech šířky hlavního okna.
Zobrazení objektů
Zde je možné nastavit způsob vykreslení geometrických útvarů. Toto nastavení se týká pouze těch objektů, který nejsou v programu ovlivněny příkazy `videt`, `styl`, `sek`, `barva` nebo `rozdel`.
Důležité je barevné odlišení objektů vybraných v *Inspektoru objektů*.
Protože tiskárny mají vyšší rozlišení než monitor, bývají čáry na vytisknutých rysech velmi tenké. Toto lze vylepšit nastavením jejich šířky na této stránce.
Popis objektů
Při popisu objektů je třeba použít tří druhů písma: pro normální popis, pro indexy a pro řecká písmena (pokud jsou názvy objektů jako alfa (– malá alfa), Omega (– velká omega), omega (– malá omega)). Můžete nastavit font a velikost v centimetrech. S programem Deskriptivní geometrie jsou dodávány dva fonty *Switzerland*, které jsou pro popisy vhodné, neboť jsou tenké a úzké.
Lze nastavit maximální vzdálenost popisu od objektu.
Papír

Tato nastavení doporučujeme nastavit raději příkazem `okraj`.

Formát čísel

Program Deskriptivní geometrie vypisuje čísla ve dvou případech: v Inspektoru objektů a v Okně zpráv. U obou může nastavit počet všech míst a počet míst desetinných.

Ostatní

U časově náročnějších postupů je vhodné vidět, jak zpracování postupu probíhá. Toto lze zajistit zobrazením informačního okna, které se může automaticky schovat po skončení zpracovaní.

Tisk

Tisknout je možné postup nebo rys. Příkazy se nacházejí v menu *Soubor*. Po zvolení těchto příkazů se objeví standardní dialogové okno pro tisk, kde je možné nastavit tiskárnu, na kterou se bude tisknout, případně její vlastnosti, zda půjde o tisk do souboru a počet kopií.
Při nastavování velikosti papíru v oknech *Tiskárna – vlastnosti* nebo *Nastavení tisku*, si uvědomte, že toto nastavení je jen dočasné. Trvalé nastavení je možné pomocí příkazů uvedených v postup (viz kapitola Práce s programem – Nastavení velikosti rysu).

Spuštění programu

Postup, který jste napsali, můžete buď provést najednou (*Postup – Spuštění* [F9]) nebo po jednotlivých krocích (*Postup – Krok / Pauza* [F8]). Příkaz *Krok* [F8] dělá kroky jen v aktivním bloku; pokud chceme projít vnořený blok musíme použít příkaz *Krok dovnitř* [Shift+F8]. Při provádění postupu po krocích je třeba překreslit celý rys po příkazech `videt, sek, rozdel, styl, popis, barva, vrat`; po ostatních dojde jen k dokreslení nových objektů. Provádění postupu je možné ukončit příkazem *Postup – Ukončení* [Ctrl+F2].
Při ladění postupu může použít speciální příkaz jazyka DG – příkaz `stop()`, který zastaví program podobně jako příkaz *Pauza* [F8], ale jde o příkaz volaný z postupu.
Při spuštění postupu příkazem *Postup – Spuštění* [F9] nedochází k spuštění celého programu od začátku, dojde pouze k provedení nově přidaných řádků. Pokud chcete provést program od začátku použijete příkaz *Postup – Nové přeložení* [Ctrl+F9].
Po provedení programu je možné sledovat průběh postupu příkazy *Postup – Krok dopředu* [F7] a *Postup – Krok dozadu* [F6].

Popis programovacího jazyka

Jak už bylo řečeno jazyk vychází z jazyka C, ale byl pozměněn pro potřeby deskriptivní geometrie, v dalším textu bude označován jako jazyk DG.
Jedná se procedurální jazyk. Program se skládá z funkcí, mezi nimiž má výhradní postavení funkce `hlavni`, která se spouští jako první, je také jedinou funkcí, kterou musí obsahovat každý program
Deklarace funkce:
identifikátor_funkce `(`*[parametr1[, parametr2...]]*`) {`
  [příkazy]
`}`
(… tři tečky znamenají, že mohou následovat další, [] tyto závorky znamenají, že to co mezi nimi není povinné)
identifikátor může obsahovat znaky anglické abecedy, číslice (nesmějí být na začátku), také znak _ (podtržítko), ale nesmí obsahovat české znaky, symboly (tečka, čárka…) nebo číslici na začátku.

Proměnné

Práce s proměnnými představuje asi největší rozdíl od ostatních programovacích jazyků. Proměnnou je v jazyku DG *skupina* objektů. Skupina nemusí obsahovat žádný objekt, ale může jich obsahovat i více. Objektem se rozumí např. číslo, text, bod, přímka… V jedné skupině mohou být i objekty různých typů.

```
a = 1; // vytvoření nové skupiny a přidání objektu typu číslo do této skupiny
a = 2; // přidání nového objektu do skupiny
```

přístup k jednotlivým objektů skupiny: `skupina:číslo`

```
b = a:0 + a:1; // sečtení prvního objektu ve skupině a s druhým
c = a + b;  // sečtení skupiny a (obsahuje dva objekty typu číslo) a skupiny b (obsahuje jeden objekt
            typu číslo), výsledek c bude skupina o dvou číslech (a:0 + b:0, a:1 + b:0)
d = a + c;  // a a c obsahují dva objekty, výsledek bude skupina o čtyřech objektech (a:0 + c:0,
            a:0 + c:1, a:1 + c:0, a:1 + c:1)
```

Využití

- zpracování více objektů najednou
- stejně pojmenované objekty (liší se jen v čísle – na rysu odlišeno buď indexem, nebo čárkami)
- funkce, které vracejí více řešení (např. průnik přímky a kružnice…)

Příkazy

Základními stavebními kameny každého programu jsou příkazy. Jazyk DG obsahuje příkazy přiřazení, volání funkce, strukturované příkazy: podmínky a cykly.
Každý příkaz musí být ukončen středníkem `;`.

Přiřazení

Syntax:
levá_skupina = pravá_skupina;
Tento příkaz přidá do skupiny na levé straně objekty v skupině na pravé straně. *Levá skupina* musí být pojmenována, na rozdíl od *skupiny pravé*, která může vzniknout jako výsledek nějakých operací a nemusí mít jméno. Takže levá skupina může už existovat (jako výsledek předcházejícího příkazu přiřazení) nebo se jedná o novou skupinu, která je vytvořena právě při tom to příkazu. Jako každý jiný příkaz je ukončen středníkem.

```
a = 5; // vytvoření nové skupiny pojmenované a a přidání objektu do této skupiny
a = 4; // přidání nového objektu do již existující skupiny
```

Výsledkem příkazu přiřazení je hodnota, která může být dál použita:

`c = b = 5;` // do skupiny `b` se přidá nový objekt a do skupiny `c` se přidá skupina `b`
c = (b = 5) + 1; // do skupiny `b` se přidá nový objekt a do skupiny `c` se přidá skupina `b` zvětšená o 1.

G *Co nejde:*

`c + 1 = 5;` // skupina na levé straně nemá jméno, vznikla v průběhu příkazu, nelze tedy do ní přiřadit
`c:0 = 5;` // obdobně jako předchozí řádek, pokud chcete přepsat objekt ve skupině použijte funkci `prepis`

Blok příkazů

Syntax:

```
{
  příkazy;
}
```

nebo `{` *příkazy;* `}`

Smyslem příkazů je to, že příkazy v tomto bloku jsou vnímány jako jeden příkaz. Použití viz podmínky a cykly.

Podmínka

Syntax:

`kdyz (`*logický_výraz*`)` *příkaz1;*
`kdyz (`*logický_výraz*`)` *příkaz1;* `jinak` *příkaz2;*

logický_výraz – tím je myšlena jakákoliv skupina, podmínka bude vyhodnocena tolikrát, kolik je v této skupině čísel. Pokud se logický výraz vyhodnotí jako číslo různé od nuly, bude proveden *příkaz1*, jinak se neprovede nic (varianta bez `jinak`), nebo se provede *příkaz2* (varianta s `jinak`).
Příkazem může být jednotlivý příkaz, nebo blok příkazů.

Cyklus

Syntax:

`dokud (`*logický_výraz*`)` *příkaz;*

příkaz se bude opakovat tak dlouho, dokud platí podmínka definovaná logickým výrazem.
V souvislosti s tímto příkazem je vhodné používat příkazy `prepis`, `zvetsi`, `zmensi`, které mění objekty ve skupině.

Příklad:

```
i = 1;
dokud (i <= 5) {
  …
  zvetsi(i); nebo zvetsi(i, 2); nebo zmensi(i); nebo prepis(i, i+0.5);
// +1              +2                –1                  +0,5
}
```

G*Pozor:* příkaz `i = i + 1` nezvětší `i` o 1, ale přidá do skupiny číslo `i` zvětšené o 1.

Funkce v jazyce DG

Existují dva typy funkcí:

1. vracející hodnotu (tedy skupinu), která může být použita v dalším příkazu
1. nevracející hodnotu, jde vlastně o skupinu příkazů, které se však od bloku liší tím, že mohou přijímat parametry

Volání funkcí:

identifikátor_funkce `(`*[parametr1[, parametr2...]]*`)`

Pokud funkce nevyžaduje žádné parametry nebo jí nechcete žádné předat, musí přesto její volání obsahovat `()`, tyto dva znaky odlišují funkci od proměnné (může existovat funkce a skupina se stejným jménem).

Deklarace funkce:

identifikátor_funkce (*[[*]parametr1[,[*] parametr2...]]*) {
  [příkazy]
}

Pokud chcete, aby funkce vracela hodnotu, musí obsahovat příkaz `vrat`, který ukončí funkci a vrátí hodnotu. *Syntax příkazu* `vrat`*:*

`vrat` *skupina;*

Tento příkaz může být ve funkci uveden i vícekrát (např. v podmínce), ale je proveden vždy jen jeden příkaz, protože po jeho provedení se funkce ukončí.

Pokud funkce tento příkaz neprovede (buď ho neobsahuje, nebo není proveden v důsledku nějaké podmínky) funkce nic nevrací, a nemůže být tedy použita ve výrazu, ale může být zavolána jako samostatný příkaz.

Ve funkci můžeme používat všechny skupiny definované ve funkci, která jí zavolala. Pokud použijeme příkaz přiřazení do takovéto skupiny, dojde však k vytvoření nové skupiny platné jen v této funkci, po jejím ukončení přestane existovat.

Znak `*` před identifikátorem parametru znamená, že tento parametr nebude předáván hodnotou, ale odkazem. To znamená, že všechny změny provedené s touto skupinou se projeví i v parametru. Odkazem mohou být předávány jen skutečné skupiny, ne výrazy.

Práce se skupinami

`prepis` přepsání skupiny nebo určitého objektu ve skupině jiným objektem

`prepis(`*skupina1, skupina2*`)` vyprázdní *skupinu1* a přidá do ní objekty ze *skupiny2*

`zrus(`*skupina*`)` zruší skupinu; pokud její objekty patří i do jiných skupin, nebudou zničeny

`m(`*objekt1, objekt2*`)` vytvoří skupinu z těchto objektů

[*objekt1, objekt2*] vytvoří skupinu z těchto objektů

Komentáře

Komentář (poznámka) je část zdrojového textu, která bude překladačem ignorována. Zpravidla se používá pro okomentování určité části programu nebo k přechodnému vyřazení části postupu.

V jazyce DG existují dva typy komentářů. První je komentář, který se zapisuje `//` a výsledkem jeho použití je ignorování zbytku řádku za `//`. Další řádky nejsou nijak ovlivněny.

Dalším typem je blokový komentář, komentář začíná `/*` a končí `*/`.

Objekt číslo

Tento objekt je vyjádřením racionálního čísla. Zapisuje se v desítkové soustavě, desetinná část je oddělena `.`.

Operátory

syntax	popis operace
a + b	sčítání
a - b	odčítání
-a	unární mínus
a * b	násobení
a / b	dělení
a % b	modulo, zbytek po celočíselném dělení (při této operaci je počítáno bez desetinné části čísel)
a ^ b	umocnění
a == b	rovnost
a != b	nerovnost
a <> b	
a < b	menší
a > b	větší

a <= b	menší nebo rovno
a >= b	větší nebo rovno
`a..b`	vytvoří skupinu, která obsahuje celá čísla od `a` do `b` včetně
`zvetsi(a, b)`	zvětší číslo `a` o `b`
`zvetsi(a)`	zvětší číslo `a` o 1
`zmensi(a, b)`	zmenší číslo `a` o `b`
`zmensi(a)`	zmenší číslo `a` o 1

Funkce `zvetsi` a `zmensi` vrací hodnotu, kterou je změněná skupina.

Logické výrazy

Pomocí čísel jsou v jazyce DG vyjádřeny i logické hodnoty. Kladnou hodnotou (true) je jakékoliv číslo různé od 0, zápornou (false) jen 0. Použít lze tyto logické operace:

&	and, a, logický součin
|	or, nebo, logický součet
!	not, negace

Matematické funkce

`moc`, `odm`	mocnina a odmocnina, mají buď je jeden parametr, pak jde o druhou (od)mocninu, anebo dva parametry, první je číslo, druhé je (od)mocnitel
`sin`, `cos`, `tan`, `cotan`	
`sinh`, `cosh`, `tanh`	
`arcsin`, `arccos`, `arctan`	
`arcsinh`, `arccosh`, `arctanh`	
`log10`, `log2`, `ln`, `exp`	
`nahoda`	
`stupne`	převod radiánů na stupně
`radiany`	převod stupňů na radiány

Poslední dvě funkce jsou velice důležité pro zadávání úhlů jako parametrů funkcí. Všechny funkce požadují úhel v radiánech a také všechny funkce vracejí úhel v radiánech, a tak když bude chtít zadat úhel ve stupních jako parametr, pište *funkce*`(…, radiany(`*úhel_ve_stupních*`), …)`. Pokud chcete převést výsledek na stupňovou míru, použijte `stupne(`*funkce*`(...) )`.

Objekt text

Tento objekt použijeme při výpisu různých zpráv do *okna zpráv*, pří zjišťování vlastností objektů nebo jejich typů. Zapisuje se mezi dvojitými uvozovkami: *˝libovolný text ˝*. Délka textu je omezena je velikostí paměti.

a + b	sčítání
a == b	porovnání
a != b	porovnání
a <> b	
text + *číslo*	převedení čísla na text a jeho přičtení k textu

---

Body a vektory jsou základními pojmy analytické geometrie. Při rýsování nepotřebujeme počítat

s vektory, ale pro počítač to je jediná rozumná možnost, jak popsat prostorové útvary.
Každý bod a vektor v prostoru má tři souřadnice (x, y, z). Každou ze souřadnic je reálné číslo.

Objekt vektor

Vektor lze zadat buď třemi souřadnicemi, dvěma souřadnicemi (pak z = 0) nebo bodem, jehož souřadnice budou použity jako souřadnice vektoru. Použít lze vektorovou algebru, a řešit tak, jakýkoliv problém analyticky, místo řešení konstrukčního.

u + v	součet vektorů (součet souřadnic)
u – v	rozdíl vektorů (rozdíl souřadnic)
u * v	skalární součin
u * *cislo*	násobek vektoru (souřadnice vynásobené číslem)
u / *cislo*	podíl vektoru (souřadnice podělení číslem)
u # v	vektorový součin
-u	opačný vektor
u == v	rovnost vektorů
u != v	nerovnost vektorů
u <> v	
v(u)	velikost vektoru
rovn(u, v)	rovnoběžnost vektorů
odch(u, v)	odchylka vektorů
u."x"	výsledkem je jedna souřadnice
u."y"	
u."z"	
u.1	získání průmětu vektoru do půdorysny – 1, nárysny – 2, bokorysny – 3
u.2	
u.3	

Objekt bod

Zadává se obdobně jako vektor: 2 nebo 3 souřadnicemi nebo vektorem, ale jeho význam je zcela jiný. Bod je základním geometrickým útvarem a pomocí něho se určují další útvary. Každý bod je vykreslen v rysu a popsán svým názvem.

A + B	součet bodů
A – B	rozdíl bodů
A * *cislo*	násobek bodu (vynásobení souřadnic číslem)
A / *cislo*	podíl bodu číslem
A == B	rovnost bodů
A != B	nerovnost bodů
A <> B	
lezi(*objekt*, A)	náležení bodu geometrickému útvaru (přímka, rovina, kuželosečka)
p(*objekt*, A)	průnik bodu s geometrickým útvarem; pokud na něm leží, výsledkem je sám bod, jinak nic
A + *vektor*	posunutí bodu A o vektor
posun(A, *vektor*)	
A."x"	výsledkem je jedna souřadnice
A."y"	
A."z"	

`A.1`
`A.2`
`A.3`
	získání průmětu bodu do půdorysny – 1, nárysny – 2, bokorysny – 3

Bod může být zadán také v obecné rovině, k jeho určení je potřeba rovina a dvě souřadnice, podle kterých je možno bod jednoznačně určit.

`bod(`*rovina*`, x, y, z)` jedna ze souřadnic může být vynechána a zapsán místo ní `?`

např.: `bod(`*rovina*`, x, y, ?)`

Další způsobem je zadaní bodu, který leží na přímce v určité vzdálenosti od jiného bodu.

`bod(`*přímka*`, bod, vzdálenost`)

Objekt přímka

Základním způsobem zadání je určení přímky pomocí dvou bodů. Druhým způsobem je bodem a vektorem. Často v deskriptivní geometrii používaným zadáním je zadání půdorysem a nárysem. Posledním způsobem zadání (nepočítáme určení přímky jako výsledku jiných funkcí) je přímka ležící v dané rovině procházející daným bodem a svírající s danou přímkou v rovině ležící daný úhel.

`primka(`*bod1*`, bod2`)

`primka(`*bod*`, vektor`)

`primka(`*přímka1*`, přímka2`) zadání půdorysem a nárysem

`primka(`*rovina*`, přímka, bod, úhel`)

Obdobně jako přímku můžeme zadat polopřímku a úsečku.

`poloprimka(…)`

`usecka(…)`

Parametry u těchto funkcí jsou stejné jako u funkce `primka`.

`a == b`	shodnost dvou přímek, nezáleží na tom, zda jde o přímku, úsečku či polopřímku
`a + B`	posunutí přímky o vektor, jehož souřadnice jsou stejné jako souřadnice bodu
`a – B`	opačné posunutí
`a + u` `a – u`	posunutí přímky o vektor
`a.”A”`	první bod přímky
`a.”B”`	druhý bod přímky
`a.”v”`	vektor přímky

`a.1`
`a.2`
`a.3`
	získání průmětu přímky do půdorysny – 1, nárysny – 2, bokorysny – 3
	výsledkem může být buď přímka, nebo bod

Objekt rovina

Rovina může být zadána:

a) třemi body neležícími v přímce `rovina(A, B, C)`

b) přímkou a bodem neležícím na této přímce `rovina(a, B)`

c) dvěma různými přímkami `rovina(a, b)`

d) `rovina(x, y, z)`, kde číslo `x` (resp. `y`, `z`) je průsečík roviny a osy `x` (resp. `y`, `z`)

e) `rovina(a, b, c, d)` – obecná rovnice: $ax + by + cz + d = 0$

`Ro == Sigma`	shodnost dvou rovin

`Ro.1`
`Ro.2`
`Ro.3`
	stopy roviny, jde o průnik rovin s 1 – půdorysnou, 2 – nárysnou, 3 –

	bokorysnou
`Ro."v"`	vektor roviny
`Ro."a"`	získání koeficientů z obecné rovnice roviny
`Ro."b"`	
`Ro."c"`	
`Ro."d"`	

Kuželosečky

Na střední škole se v předmětu deskriptivní geometrie vyučují kuželosečky. Rýsování kuželoseček – elips, hyperbol a parabol je obtížné a proto se musíme při ručním rýsování spokojit s přibližnými metodami rýsování těchto křivek. Výhody programu Deskriptivní geometrie se v této oblasti projeví ještě výrazněji než při rýsování jednodušších objektů.
V programu je každá kuželosečka reprezentována analyticky – konkrétně obecnou algebraickou rovnicí druhého stupně v *x* a *y*. Taková kuželosečka může být zadána v půdorysně nebo nárysně nebo v rovinách s nimi rovnoběžnými. Při vykreslení kuželosečky se bere v úvahu nastavení zobrazení stejně jako u ostatních objektů
- tj. změna síly, stylu a barvy čáry, rozdělení kuželosečky na více části apod.
Výjimku tvoří kružnice, která může být zadána v obecné rovině, není reprezentována obecnou rovnicí, ale rovinou, středem a poloměrem. Na kuželosečku se převádí, až když je vykreslována (jejími průměty může být kružnice nebo elipsa). Ostatní kuželosečky musí ležet v rovinách rovnoběžných s průmětnami. V tomto rozsahu je i učivo na střední škole.

Při řešení úloh v deskriptivě nemůžeme zadávat kuželosečky rovnicí, ale podle takových vlastností, ke kterým může dospět pomocí konstrukcí. Proto můžeme zadat kuželosečky několika způsoby:

- kružnici: `kruz(`*rovina*`, `*střed*`, `*poloměr*`)`

Průmětem kružnice není jen kružnice, ale i elipsa, proto je v jazyce DG kružnice dalším typem objektu, objekty typu kuželosečka jsou jen průměty této kružnice (`k.1`, `k.2`)

- `elipsa(`*rovina*`, F, G, a)` - rovina musí být rovnoběžná s půdorysnou a nárysnou, `F` a `G` jsou ohniska, `a` délka hlavní osy
- `elipsa(`*rovina*`, A, C, S)` - `A` hlavní vrchol, `C` vedlejší vrchol, `S` střed
- `hyperbola(`*rovina*`, F, G, a)`
- `parabola(`*rovina*`, F, d)` - ohnisko `F` a řídící přímka `d`

Na tyto typy zadání lze převést jakékoliv jiné zadání a zadat tak jakoukoliv kuželosečku.
V knihovně `"kuzelosecky.dk"` najdete konstrukce sestrojení tečen z bodu nebo daným směrem. Bližší popis viz tato knihovna.

Lze určit průnik dvou kuželoseček a průnik kuželosečky s přímkou. Výpočet průniku dvou kuželoseček je řešením dvou rovnic vyjadřujících kuželosečky. Tato soustava může nabývat žádného, jednoho až čtyř, nebo nekonečně mnoho řešení. Nalézt je není možné pomocí matematických úprav, proto se používá numerických metod. Mezi další operace patří test náležení bodu kuželosečce, posunutí a otočení kuželosečky. Průnik roviny a kuželosečky je vlastně průnikem kuželosečky a průsečnice této roviny a roviny kuželosečky.

Pokud se jedná o středovou kuželosečku (kružnici, elipsu a hyperbolu), lze zjistit její střed pomocí operátoru . a vlastnosti "S".
`S = k."S"`
Další vlastnosti:
`"ro"` – rovina, ve které kuželosečka leží

"typ" – druh kuželosečky: elipsa, hyperbola, parabola
"u" – odchylka osy
"DD", "dd" – diskriminant kuželosečky a diskriminant kvadratických členů
"a11" – "a33" – koeficienty v rovnici
u elips a hyperbol:
"F", "G" – ohniska
"A", "B" – hlavní vrcholy
"C", "D" – vedlejší vrcholy
"h", "v" – hlavní a vedlejší osa
"a", "b", "e" – velikost hlavní a vedlejší poloosy, excentricita
u parabol:
"F" – ohnisko
"d", "v" – řídící přímka, vrcholová tečna
"V" – vrchol
"p" – velikost parametru
"o" – osa

Kuželosečky je možné posouvat a otáčet pomocí funkcí `otoc` a `posun`, výsledkem těchto funkcí je transformovaná kuželosečka.

```
f = posun(e, vektor(2, 5, 6));
r = otoc(p, radiany(45));
```

Příklady:

zadání kuželosečky vyjádřené rovnicí $9x^2 + 48xy - 36x + 2y^2 - 24x + 21 = 0$ v půdorysně:

```
k = kuzelosecka(Pi, 9, 24, -18, 2, -12, 21);
```

- kružnice:

```
Ro = rovina(5, 7, 8);
S = bod(Ro, -4, 3, ?);
k = kruz(Ro, S, 4);
```

- elipsa, hyperbola a parabola:

```
e = elipsa(Pi, F, G, 5);
h = hyperbola(Ny, F, G, 6);
p = parabola(Pi, F, d, p);
```

Knihovna "kuzelosecky.dk" obsahuje i Rytzovu konstrukci elipsy zadané dvěma sdruženými průměry.

Práce s programem

Nastavení velikosti rysu

Již před psaním postupu by mělo být jasné, jaké velikosti bude papír, na kterém bude vykreslen rys. K nastavení velikosti papíru patří i výběr jeho orientace (vodorovné nebo svislé) a nastavení okrajů, případně i rámečku, jenž bude narýsován kolem rysu.
Tyto nastavení lze provést dvěma způsoby:

1. Provede je sám uživatel v dialogovém okně *Nastavení tisku* a *Nastavení programu.* Tato nastavení budou nejen pro aktivní postup, ale pro všechny ostatní.
1. Nastavení jsou zapsána do postupu, a tak každý postup má svoje nastavení, které se projeví při jeho spuštění.

Způsob 2. je pro nás výhodnější, když používáme speciální velikost. Pokud používáme obvyklou velikost A4 a svislou orientaci, nemusíme toto nastavení zapisovat do postupu.
`papir(`*výška*`,` *šířka*`,` *orientace* [`,` *měřítko*]`)`; // *výška* a *šířka* jsou velikosti papíru v centimetrech, *orientace* může být buď 1 (svisle), nebo 2 (vodorovně). Měřítko udává velikost jednoho centimetru na rysu.
`okraj(`*horní*`,` *dolní*`,` *levý*`,` *pravý*`,` *vnitřní*`,` *tloušťka_čáry*`,` *styl_čáry*`)` // velikost okrajů v centimetrech, vnitřním okrajem je myšlena vzdálenost mezi kreslenými útvary a okrajovou čáru, na kterou nebude kresleno. *tloušťka_čáry* může nabývat hodnot od 0 výše. *styl_čáry* může být 0 – plná, 1 – čárkovaná, 2 – čerchovaná, 3 – tečkovaná, 4 – čárka-tečka-tečka.

Výběr zobrazovací metody

Základní zobrazením používaným v programu je Mongeovo promítání. Toto promítání je nastaveno, aniž musíte něco psát. Mezi další promítání, které program (bude) ovládá je kótované promítání a zobrazení jen půdorysu bez kót (jde vlastně jen obyčejný dvourozměrný papír). Další možné zobrazovací metody jako kosoúhlé promítání, axonometrie a perspektiva jsou náměty pro rozšíření programu v budoucnosti.

```
zobrazeni("mongeovo")      // nastaví Mongeovo promítání
zobrazeni("kotovane")
zobrazeni("papir")
```

Důležitým nastavením je posun souřadnicové soustavy. To umožňuje příkaz `posunuti(x, y)`.

```
posunutipps(2, -2)         posune počátek souřadnicové soustavy dva centimetry doprava a nahoru.
```

Použití knihoven

Ve složitějších postupech se opakují nejen základní konstrukce, ale i další postupy, které jsou stejné, liší se pouze objekty, se kterými se provádí. Protože by bylo zbytečné psát podobné postupy vícekrát, můžeme naprogramovat funkci, která provede tento postup se svými parametry, které mohou být při každé jejím volání jiné.
Některé takto vytvořené funkce jsou vhodné i pro jiné postupy, a proto je vhodné umístit do jiného souboru – *knihovny*, která může být použita při psaní každého postupu. Toto zajistíme pomocí příkazu
`#vloz "soubor_s_knihovnou.dk"`.
A jak vypadá knihovna: na rozdíl od obyčejného postupu neobsahuje funkci `hlavni`, ale obsahuje funkce, které mohou být použity v jiném postupu. Knihovna může také používat funkcí z jiných knihoven, kromě sebe samé. Při používání více knihoven hrozí nebezpečí, že některé funkce se stejným názvem budou obsaženy ve více knihovnách. V tom případě ohlásí program chybu a uživateli nezbývá nic jiného, než funkce přejmenovat.

Způsob vykreslení objektů

Způsob vykreslení geometrických útvarů – velikost, tloušťka, barva, styl bodů a čar – jsou důležité pro přehlednost rysu. Např. důležité čáry rýsujeme tlustší, pomocné konstrukce slaběji, jiný styl čar použijme, když je čára viditelná (použije plnou čáru), nebo když není vidět (použijeme čárkovanou čáru), čerchovaně kreslíme např. osy těles nebo konstrukce v otočení. Akce nastavení probíhá ve dvou úrovních:

1. standardní nastavení pro všechny rysy a postupy v dialogovém okně *Nastavení programu*
1. nastavení jen některých objektů pomocí příkazů uvedených v postupu

Způsob vykreslení může být jiný v půdorysu i v nárysu.

`videt(`*objekt*`,` [*část*`,`] *plán*, *viditelnost*`,` [[*velikost_bodu*`,` *styl_bodu*], *tloušťka_čáry*`,` *styl_čáry*])

plán `1` – půdorys, `2` – nárys, `vse` – půdorys i nárys

část část objektu, kterou bude platit nastavení, pokud je vynechána platí první část, je možné použít identifikátor `vse` pro nastavení všech částí

viditelnost 0 – nejde vidět, 1 – je vidět jen v kladné části souřadnicové soustavy, 2 – jen v záporné, 3 – v celé souřadnicové soustavě

velikost_bodu a *styl_bodu* jsou nastavení nejen pro bod, ale i pro přímku, protože speciálním přímky zobrazením může být bod. Pokud je chcete ponechat v předešlém stavu napište místo hodnot `?`

velikost_bodu je zadána v centimetrech a vyjadřuje poloměr jeho velikosti, pokud je velikostí 0, bude vykreslen jako jeden *pixel* (nejmenší zobrazitelný pod na monitoru nebo tiskárně); velikost –1 určuje, že bod nebude vidět – to je důležité u malých čar (například vniklých spojením bodů z bodové konstrukce, které jsou blízko u sebe)

styl_bodu 0 – křížek **+**, 1 – průhledné kolečko ° (je pod ním vidět čára), 2 – bílé kolečko °, 3 – barevné kolečko •

tloušťka_čáry vyjadřuje velikost v pixelech, může nabývat hodnot od 0 výše

styl_čáry 0 – plná, 1 – čárkovaná, 2 – čerchovaná, 3 – tečkovaná, 4 – čárka-tečka-tečka.

`barva(`*objekt*`,` [*část*`,`] [*plán*`,`] *název_barvy*)

název_barvy text s názvem barvy, “`cerna`“, “`bila`“, “`modra`“, “`zelena`“, “`cervena`“, “`zluta`“, “`seda`“

Velmi důležitou vlastností je rozdělení jednoho objektu na více částí, které pak jsou vykreslit jiným způsobem. Rozseknutí útvaru se děje pomocí několika přímek, každá přímka vymezuje kladnou polorovinu, ve které smí ležet body objektu. Pokud potřebuje zápornou polorovinu, stačí zapsat body v opačném pořadí. Zadaní jedné podmínky zapíšeme příkazem `sek`, zapsání dalších podmínek pro stejnou část dosáhneme přidáním další příkazu `sek`.

`sek(`*objekt*`,` [*část*`,`] *plán*`,` *A*`,` *B*`,` *znaménko*)

část nastavení části, ke které se příkaz vztahuje, platí stejné podmínky jako u předešlých příkazů (pokud je vynechána platí první část, je možné použít identifikátor `vse` pro nastavení všech částí)

A, B body určující přímku a polorovinu (přímka je převedena do obecné rovnice, a polorovina je pak určena takto: $ax + by + c > 0$). Z toho vyplývá, že pokud zapíšeme body v obráceném pořadí, vyjádříme tak opačnou polorovinu.

znaménko tento parametr ovlivňuje nastavení celé části (ne pouze jedné podmínky), rozhoduje poslední příkaz, který tuto hodnotu mění. Může nabývat hodnot 0 nebo 1. Pokud je nastaven na 1, je vykresleno to, co odpovídá podmínkám, když na 0, je vykresleno to, co jim neopovídá

Pomocí příkazů `videt`, `barva` a `sek` můžeme nastavit nejrůznější způsoby vykreslení, ale jejich použití je složité a zdlouhavé. V praxi je často potřeba rozdělit čáru (přímku nebo kuželosečku) na několik částí

podle bodů, které na ní leží. Způsob vykreslení se pak po částech střídá. Tím lze jednoduše nastavit viditelnost (střídají se části viditelné a neviditelné).

`rozdel(`*objekt*`,` *plán*, *tloušťka_čáry1*`,` *styl_čáry1, tloušťka_čáry2*`,` *styl_čáry2, bod1, bod2, …*`)`

tloušťka_čáry1`,` *styl_čáry1*	způsob vykreslení lichých čar
tloušťka_čáry2`,` *styl_čáry2*	způsob vykreslení sudých čar
bod1, bod2, …	body, podle kterých bude čára rozdělena

Body musí být uvedeny v pořadí, jak leží na křivce!

U kuželoseček funguje funkce téměř stejně. U kuželoseček nedochází k jejich uzavření (spojení posledního a prvního bodu), lze tak zobrazit jen část křivky. Pokud mezi dva body napíšete jako parametr číslo 1, budou tyto dva body spojeny delším obloukem, jinak se spojí kratším obloukem.

Popis objektů

Popis objektů je nezbytnou součástí programu, protože umožňuje orientaci ve složitějších rysech. Každý objekt, který je zobrazen je popsán podle obvyklých pravidel pro popisování. Každý objekt je popsán svým identifikátorem a svým číslem ve skupině. Číslo ve skupině je napsáno jako horní index. Při počtu objektů ve skupině do tří (čísla 0, 1, 2), jsou číselné indexy nahrazeny čarami. U prvního objektu ve skupině je index vynechán. Písmo a velikost lze nastavit v okně *Nastavení…*.

V případě, že několik objektů je vykresleno na stejné místo, projeví se to i v popise, kdy takový objekt popsán jedním popisem ve formátu `A=B=C`.

Popis v žádném případě nesmí překrývat jiný popis nebo bod. Pokud je to možné, nepřekrývá ani čáry. Vhodné (prázdné) místo pro popis ve složitějších rysech je někdy daleko od původního místa, a tak je vhodnější umístit popis sice přes čáru, ale na bližší místo.

Rozdíly jsou v popisu v různých promítáních:

- v Mongeově promítání je objekt zobrazen dvakrát – v půdorysu a nárysu –, a tudíž je nutné každé zobrazení zvlášť popsat – odlišují se dolními indexy za názvem.
- v kótovaném promítání se u bodů přidává kóta – závorka s třetí souřadnicí bodu

Konstrukce

Konstrukce jsou vlastně funkce, které vytvářejí z několika vstupních objektů (parametrů) objekt, který je výsledkem této konstrukce. Mnoho z následujících konstrukcí jde zapsat pomocí několika jiných příkazů, ale program je přesto ovládá z důvodu jednoduššího použití (stačí napsat jeden příkaz) a větší rychlosti (v programu DG jsou naprogramovány většinou podle vztahů analytické geometrie, kterou je počítač schopen rychleji a přesněji zpracovat).

Bod ležící v rovině

Bod nemusí být zadán právě třemi souřadnicemi, ale tím, že leží v určité rovině a jeho dvě souřadnice jsou zadané, ta třetí bude dopočítána (v případě, že takové zadání jednoznačně určuje bod).

`bod(`*rovina*`,` *x, y, ?*`)` nebo `bod(`*rovina*`,` *x, ?, z*`)` nebo `bod(`*rovina*`,` *?, y, z*`)`

Rovnoběžky a kolmice

přímka rovnoběžná s přímkou vedená bodem

`rovnobezka(`*přímka*`,` *bod*`)`

přímka v rovině rovnoběžná s přímkou v určité vzdálenosti od této přímky

`rovnobezka(`*rovina*`,` *přímka*`,` *vzdálenost*`)`

přímka v rovině kolmá ke přímce vedena bodem

`kolmice(`*rovina*`,` *přímka*`,` *bod*`)`

přímka kolmá ke přímce vedená bodem

`kolmice(`*přímka*`,` *bod*`)`

přímka kolmá k rovině vedena bodem
`kolmice(`*rovina, bod*`)`

Rovnoběžné a kolmé roviny

rovina rovnoběžná s rovinou vedená bodem
`rovnobezna(`*rovina, bod*`)`
rovina kolmá k přímce vedená bodem
`kolma(`*přímka, bod*`)`

Sklápění a otáčení

Tyto dvě konstrukce patří mezi základní operace, bez kterých se neobejde většina řešení úloh středoškolské deskriptivní geometrie.

Tyto konstrukce lze naprogramovat v pomocí jiných konstrukcí, které už v jazyce DG existují. Najdete je v knihovně `monge_sklop.dk`, jsou určený pro Mongeovo promítání. Jde však o často používanou konstrukci proto je naprogramována i v jazyce DG, aby její provádění bylo co nejrychlejší. Jde o funkce `otoceni`, `otoceniz`, `notoceni`, `notoceniz`. (`z` znamená zpět, `n` nárysna)
Funkce `otoceni` a `notoceni` mají dva parametry: skupina objektů (bodů) a hlavní přímku I. osnovy roviny, kolem které se bude bod otáčet. Výsledkem je skupina otočených objektů.
Funkce `otoceniz` a `notoceniz` mají opačnou funkci: bod ležící v půdorysně (nárysně) otočit zpět do roviny. Požadují tři parametry: skupinu, která se bude otáčet, rovinu, do které se bude otáčet, a hlavní přímku I. osnovy této roviny. Výsledkem je opět skupina otočených objektů.

Funkce pro sklápění jsou obdobné: `sklopeni`, `sklopeniz`, `n sklopeni`, `nsklopeniz`.
Tyto funkce mají dva parametry: bod, který se bude sklápět, a přímku, kolem které se bude sklápět.

Průnik

`prunik(`*objekt1, objekt2*`)`

Funkce průnik funguje pro všechny základní objekty (bod, přímka, rovina, kuželosečka). Pořadí zápisu objektů má být v tomto pořadí: kuželosečka, rovina, přímka, bod. Výsledkem této funkce může být skupina jakýkoliv objektů. Pokud je nastane případ prázdného průniku, je výsledem speciální typ nic.
Funkce `prunik` řeší průnik analyticky, což je rychlejší než řešení konstrukční.

Mnohoúhelník

Často používaným útvarem je mnohoúhelník, který je zadán více než dvěma body. V programu není reprezentován jako nový typ objektu, ale jako skupina úseček. Takový mnohoúhelník lze vytvořit pomocí funkce `nuhelnik`.
`nuhelnik(`*bod1, bod2, …, bodn*`)` vytvoří skupinu úseček (první vede z *bod1* do *bod2*, druhá z *bod2* do *bod3*, poslední z *bodn* do *bod1*).

Bodová konstrukce

Řešením složitějších úloh (např. průnik dvou válců v obecné poloze) není křivka, kterou známe, umíme matematicky popsat a zobrazit. Jediné, co dokážeme pomocí konstrukcí vytvořit je množina bodů, které náleží křivce. Tyto body pak na papíře spojíme do jedné křivky. Totéž musíme udělat na počítači. Protože body vzniklé z bodové konstrukce většinou nenásledují přesně po sobě, nemůžeme použít funkce `nuhelnik`. Nejdříve musíme body uspořádat tak, aby následovali po sobě. Může nastat několik případů:

1. Pokud je výsledkem každé dílčí bodové konstrukce nejvýše jeden bod, pak tyto body většinou následují po sobě (to lze ověřit v *inspektoru objektů*) a lze je spojit funkcí `nuhelnik` nebo `ncara`.

1. Výsledkem jsou dva nebo více bodů, které sice nejdou po sobě, ale lze je tak uspořádat, protože např. z dvou bodů patří jeden do první poloviny a druhý do druhé poloviny křivky. V tomto případě je nutné vědět, jakým způsobem body vznikají a podle toho je pak přerovnat. To lze udělat pomocí několika cyklů:

```
// sudé v normálním pořadí
i = 0;
dokud (i < pocet(R)) {
  NR = R:i;
  zvetsi(i, 2);
}
zrus(i);
// liché obráceně
i = pocet(R)-1;
dokud (i >= 0) {
  NR = R:i;
  zmensi(i, 2);
}
zrus(i);
```

Pro zjednodušení používání těchto akcí je v jazyce DG připraveno několik funkcí:
`usp(`*počet_částí*`, [-]`*část1*`, [-]`*část2*`, …, `*body*`…)`

počet_částí	hodnota toho parametru je počtem bodů z jednotlivé konstrukce, rovná se počtu částí křivky
část	číslo části; pokud je záporné, bude pořadí bodů obrácené
body	jedna nebo více skupin, obsahující body

Provedení těchto zabudovaných funkcí je samozřejmě rychlejší než provedení obdobných funkcí zapsaných v jazyce DG.

1. Celkovým výsledkem je skupina bodů, které nejdou po sobě a nejde je uspořádat. Takové body lze spojit do jedné křivky pomocí funkcí `body`, `bodys`, `bodyn` nebo `bodysn` . Tyto funkce spojí jednotlivé body tak, že je vždy spojen nejbližší s nejbližším podle jejich vzdáleností v půdoryse a náryse zvlášť.

`body(`*bod1*`, `*bod2*`, …, `*bodn*`)`	vytvoří skupinu čar z bodů
`bodys`	navíc je spojen první a poslední bod
`bodyn, bodysn`	pro nárys

Příklad:
Tento příklad ukazuje jeden způsob, jak je možné provést bodovou konstrukci pomocí cyklu `dokud`.
V tomto případě se mění výška a v každé výšce se provádí tatáž konstrukce.

```
rez (z) {
  . . .            // vlastní postup jednotlivé konstrukce
  vrat X;          // X je nějaký bod
}

z = 0.5;           // počáteční výška
krok = 0.1;        // krok v cm, tento krok lze libovolně změnit (čím menší číslo, tím větší kvalita,
                   ale delší výpočet)

dokud (z <= 4.5)   // cyklus omezen maximální výškou
```

```
{
    R = rez(z);     // jednotlivá konstrukce
    prepis(z, z+krok);   // zvetšení výšky
}
```

Podle výše uvedených případů:
1. Body spojíme do jedné křivky funkcí `nuhelnik`
`cR = nuhelnik(R);` nebo `cR = ncara(R);`

2. Skupina bodů se skládá ze dvou částí, které jsou do sebe vloženy. Proto je funkcí `usp` přerovnáme:
```
NR = usp(2, 1, -2, R);
cR = nuhelnik(NR);   nebo    cR = ncara(NR);
```

3. Skupina `R` obsahuje řadu bodů, ty spojíme do jedné křivky pomocí funkce `bodys`:
```
cR = bodys(R);      // půdorys
cR2 = bodysn(R);    // nárys
```

Toto platí pro všechny tři případy:
```
videt(R, vse, 0); // schováme původní body
videt(cR, 1, 1, 1, 0, 0, 2, 0);      // část křivky cR v kladné části silnější plnou čarou
videt(cR, 2, 1, 3, 0, 0, 1, 3);      // část křivky cR v záporné části tečkovanou čarou
```

Analytická geometrie
Program DG obsahuje i funkce známe z analytické geometrie. Dají se rozdělit do dvou skupin:
1. zjistitelné na papíře hned pomocí pravítka nebo úhloměru – jedná se o útvary v rovině rovnoběžné s půdorysnou či nárysnou
2. k jejich zjištění je provést určité konstrukce – útvary v prostoru

Funkcím je jedno, zda se jedná o případ 1. nebo 2.
Funkce 2. používejte v případě, že umíte provést obdobnou konstrukci a chcete použít funkci, která je zabudovaná v programu, a je tedy rychlejší.

Velikost
`v(`*`vektor`*`)` velikost vektoru

Vzdálenost
Výsledkem je vzdálenost.
`v(`*`bod`*`, `*`bod`*`)` vzdálenost dvou bodů
`v(`*`přímka`*`, `*`bod`*`)` vzdálenost bodu od přímky
`v(`*`rovina`*`, `*`bod`*`)` vzdálenost bodu od roviny

Rovnoběžnost
Výsledem je logická hodnota.
`rovn(`*`přímka`*`, `*`přímka`*`)` jsou přímky rovnoběžné?
`rovn(`*`rovina`*`, `*`přímka`*`)` je rovina rovnoběžné s přímkou?
`rovn(`*`rovina`*`, `*`rovina`*`)` jsou roviny rovnoběžné?
`rovn(`*`vektor`*`, `*`vektor`*`)` jsou vektory rovnoběžné?

Kolmost
Tuto metrickou vlastnost lze popsat pomocí odchylky. Pokud je odchylka 90, jsou útvary k sobě kolmé.

Odchylka
Výsledkem je úhel v radiánech.

`odch(`*přímka*`, `*přímka*`)`
`odch(`*rovina*`, `*přímka*`)`
`odch(`*rovina*`, `*rovina*`)`
`odch(`*vektor*`, `*vektor*`)`

Vyplňování oblastí

Pro vyplňování oblastí byl zvolen poměrně jednoduchý způsob zadání plochy. Zadáte okrajové křivky (jakýkoliv objekt – přímka, kuželosečka…) a jeden bod. Vybarvena bude oblast kolem tohoto bodu, dokud nebude dosáhnuto hraniční křivky. Jde o podobný jako u grafických programů.
Existují dvě funkce vypln a vypln2. Rozdíl je v tom, že funkce vypln vykreslí všechny čáry plně a tence, zatímco vypln2 ponechá jejich styl.
Syntax:
vypln[2](*bod*, *plán*, *barva*, *styl_výplně*, *ohraničení*)
Pokud chcete uvést více bodů nebo více křivek do ohraničení, použijte vytvoření nové skupiny:
vypln([A, B], 1, RGB(100, 200, 50), 0, [a, b, c]);
styl_výplně může být 0 – plný, 1 – 6 různé druhy šrafování: čárky vodorovné, svislé, 45, 135, mříž, mříž otočená o 45. Výplně jsou modifikovatelné, nacházejí se v adresáři Výplně. Rozlišují se na výplně pro tiskárnu (s větším rozlišením) a pro monitor (s nižším rozlišením). Maximálně může vytvořit až 100 různých výplní.

Jiné promítací metody

V programu Deskriptivní geometrie verze 1.0 je zabudovány dvě promítací metody, které zobrazují třírozměrné útvary: kótované promítání a Mongeovo. To neznamená, že nemůžete vytvořit rys v axonometrii nebo jiném promítání. Lze to řešit stejným způsobem, jako kdyby se rýsovalo na papíře. Nastavíte zobrazeni(“papir“). Pokud chcete vytvořit bod, nemůžete použít funkci bod(x, y, z), ale je třeba napsat postup, kterým vytvoříte bod ve vašem promítání. Je vhodné tento postup zapsat jako funkci.
K programu je dodávána knihovna “axonometrie.dk”, která pracuje na uvedeném principu. Je před začátkem rýsování je třeba nastavit axonometrii: velikost axonometrického trojúhelníku, osy, sklopené osy; tato nastavení jsou používána dalšími funkcemi jako je abod(x, y, z), která vytvoří dva body: axonometrický průmět bodu o zadaných souřadnicích a jeho půdorys.
Knihovna obsahuje i další konstrukce pro axonometrii, nedělá si však nárok na úplnost. Jde jen o příklad toho, jak lze program rozšířit na uživatelské úrovni.