Rajzoló automata

Az alábbiakban egy rajzoló automata leírását olvashatjuk. A feladatok megoldása során a későbbiekben ezzel az automatával kell dolgoznunk.

Vízszintes asztallapon nagyméretű rögzített papírlap helyezkedik el. Egy automata rajzoló karja a papírlap felett mozoghat, annak minden egyes pontját eléri és oda egy pontot képes rajzolni.

Az automatának utasításokat adhatunk, amelyeket abban az esetben képes végrehajtani, ha formájukban és tartalmukban az utasítások hibamentesek. Ellenkező esetben az automata hibaüzenetet küld, és nem hajtja végre a hibás utasítást, addig várakozik, míg ki nem javítjuk a hibát.

Az utasítások formailag egy utasításnévből és egy ezt követő zárójelpárból állnak. A zárójelekben az utasítások egy vagy több paramétere szerepel, több paraméter esetén pontosvesszővel elválasztva, melyeket kötelező megadni (pl.: *RajzolObjektumVéletlenPont(obj), RajzolEgyenes(e;A;B)*). A paraméterek síkbeli objektumokat jelölnek, melyeket formailag tetszőleges nem üres karaktersorozattal (string) nevezhetünk meg. Az egyes paraméterek sorrendjének és típusának megfelelőnek kell lennie. A *RajzolEgyenes(e;A;B)* eljárásban például *A* és *B* nyilvánvalóan nem jelölhet ponttól különböző objektumot.

Az automata hét különböző utasítást ért meg, melyek két típusba sorolhatók: öt rajzoló eljárás, két technikai eljárás.

Rajzoló eljárások:

Ha egy rajzoló eljárás végrehajtható, akkor egyrészt megjelenik az alakzat rajzolata a rajzpapíron, másrészt eltárolódik az automata memóriájában az alakzat neve és a hozzá tartozó paraméterek. Ettől kezdve az automata képes a megnevezett alakzattal dolgozni.

RajzolObjektumVéletlenPont(obj)
kirajzolja az *obj* nevű objektum egy véletlenül kiválasztott pontját; az objektumnak korábbról már ismertnek (definiáltnak) kell lennie! Mivel többször is kiadhatjuk ezt a parancsot ugyanarra az objektumra, az objektum több véletlen pontja is tárolva lesz a memóriában. Ezért az automata a létrehozás sorrendjében sorszámozza a véletlen pontokat *vP<n>obj* névvel (pl. *vP1e*, *vP2e* egy korábban már definiált *e* nevű egyenes egymást követően elsőként és másodikként kiválasztott véletlenpontjai); ha az *obj* nevű objektum nem létezik az eljárás hibaüzenetet küld, nem hajtódik végre; A rajzlap *LAP* néven alapértelmezett objektum.

RajzolEgyenes(e;A;B)
megrajzol egy egyenest két korábban már megrajzolt *A* és *B* ponton át és elnevezi ezt az egyenest *e*-nek; ha *A* és *B* pontok valamelyike nem létezik az eljárás hibaüzenetet küld, nem hajtódik végre

RajzolSzakasz(r;A;B)
megrajzol egy egyenes szakaszt *A* és *B* végpontokkal és elnevezi *r* -nek; ha *A* és *B* pontok valamelyike nem létezik az eljárás hibaüzenetet küld, nem hajtódik végre

RajzolKör(k;O;r)
körvonalat rajzol az *O* pont körül az *r* szakasz-objektummal, mint sugárral és elnevezi ezt a körvonalat *k*-nak; az *O* pontnak és az *r* szakasz-objektumnak korábbról már ismertnek (definiáltnak) kell lennie, egyébként az eljárás hibaüzenetet küld, nem hajtódik végre!

RajzolMetszet(obj1;obj2)
kirajzolja az *obj1* nevű és az *obj2* nevű síkbeli objektumok közös pontját vagy pontjait *m<n>obj1_obj2* néven, ahol *n* a megrajzolás sorrendje szerinti sorszámot jelenti; az objektumoknak korábbról már ismertnek (definiáltnak) kell lenniük!; ha a metszéspontok száma végtelen vagy 0, akkor egyetlen speciális nevű pont objektum tárolódik a memóriában, mely nem rajzolódik ki!; végtelen sok metszéspont esetén m1obj1_obj2 neve: □□, 0 metszéspont esetén m1obj1_obj2 neve: □ ; ha az *obj1* és *obj2* nevű síkbeli objektumok valamelyike nem létezik az eljárás hibaüzenetet küld, nem hajtódik végre.

Technikai eljárások:

TörölObjektumVéletlenPont(obj)
törli a rajzlapról és a memóriából az *obj* nevű objektum összes korábban rögzített véletlenpontját.

ElnevezPont(Q,R)
Elnevez egy már létező *Q* nevű pontot *R* nevűnek, ezt követően mindkét pontra lehet hivatkozni, a pontok azonosan egyenlők; ha *Q* pont nem létezik az eljárás hibaüzenetet küld, nem hajtódik végre

Mintapélda:

Az alábbi három, lényegében azonosnak tűnő eljárás egyenest (*e*) igyekszik rajzolni két véletlenül választott ponton át (*A*,*B*):

1 RajzolObjektumVéletlenPont(Lap)
RajzolObjektumVéletlenPont(Lap)
RajzolEgyenes(e; vP1Lap, vP2Lap)

2 RajzolObjektumVéletlenPont(Lap)
RajzolObjektumVéletlenPont(Lap)
ElnevezPont(vP1Lap, A)
ElnevezPont(vP2Lap, B)
RajzolEgyenes(e; A, B)

3a *Eljárás RajzolEgyenes*
> <
Ciklus
RajzolObjektumVéletlenPont(Lap)
RajzolObjektumVéletlenPont(Lap)
amíg nem PontAzonosság[(1)](vP1Lap; vP2Lap)
ElnevezPont(vP1Lap, A)
ElnevezPont(vP2Lap, B)
TörölObjektumVéletlenPont(Lap)
RajzolEgyenes(e; A, B)
Eljárás vége

Az *első* algoritmus a szokásos geometriai szerkesztés szemszögéből közelíti meg a feladat megoldását, a *második*, talán kissé mesterkéltnek tűnő módon csinosítja a megoldást, elnevezi a véletlenpontokat. A *harmadik* algoritmus (első változata) a strukturált programozás elemeit használva, programozási nyelvtől függetlenül adja meg az algoritmus forráskódját. Ebben megjelenik egy végén tesztelős ciklus, melyből csak akkor lép ki az automata, ha a generált véletlen pontok valóban különböznek egymástól. Egy automata esetében előfordulhat ugyanis, hogy *vP1Lap* ≡ *vP2Lap.* (Egy manuálisan végzett szerkesztés során magától értetődően nem adunk meg két azonosan egyenlő pontot, hiszen tudjuk, egy pont nem határozza meg az egyenes helyzetét: az egyenes a pont körül, mint középpont körül, foroghatna.) Látható, hogy egy automatára bízott cselekvéssort kissé kritikusabban kell elemezni, ami ugyanis számunkra természetes, az nem feltétlenül természetes az automata számára is. Valójában arról van szó, hogy tudatosítanunk kell a megoldást működtető összes rejtett tartalmat, és ezeket az automata tudomására kell hoznunk.

---

(1) A *PontAzonosság(vP1Lap;vP2Lap)* egy logikai függvény (visszatérési értéke: igaz/hamis), melyet az automata által ismert eljárásokból hozhatunk létre.

A rajzoló automata aktuális nyelvi megvalósításának további következményei vannak eljárásunkra nézve. Az egyik ilyen lehetséges következmény miatt harmadik algoritmusunk első változata, a megadott formában sajnos hibás! Ha ugyanis programunk összetett, és egy korábbi utasításrészletben definiáltunk már véletlen pontokat, akkor a ciklus kilépési feltételében nem is az aktuálisan generált pontok azonosságát vizsgáljuk! Előfordulhat ugyanis, hogy a korábbiakban generált első és második véletlen pontra *vP1Lap* ≡ *vP2Lap,* így programunk hiába generálja újra és újra – emelkedő sorszám indexel - a véletlenpontokat, kilépési feltételként valójában nem ezeket hasonlítja össze. Így végtelen ciklusba kerülhet (látszólag lefagy). Továbbá, ha *vP1Lap* ≠ *vP2Lap* lenne is, nem az aktuálisan generált két új véletlen ponton át (*vPnLap, vPn+1Lap (n>2))* rajzolja az automata az egyenest, hanem a korábbiakban megadott két véletlen ponton át. A helyes megoldáshoz akkor jutunk, ha a zavart okozó esetleges korábbi véletlenpontokat idejekorán töröljük:

3b *Eljárás RajzolEgyenes*
>**TörölObjektumVéletlenPont(Lap)**<
Ciklus
RajzolObjektumVéletlenPont(Lap)
RajzolObjektumVéletlenPont(Lap)
amíg nem PontAzonosság(vP1Lap; vP2Lap)
ElnevezPont(vP1Lap, A)
ElnevezPont(vP2Lap, B)
TörölObjektumVéletlenPont(Lap)
RajzolEgyenes(e; A, B)
Eljárás vége

Ez a megoldás már majdnem jó. Sajnos azonban előfordulhat, bár kicsi eséllyel, hogy az automata által kifejezetten a készülő egyenesünk számára generált két új pont újra azonos lesz! Nos ekkor újra belefutunk a fentebb említett problémába! A helyes megoldás:

3c *Eljárás RajzolEgyenes*
Ciklus
TörölObjektumVéletlenPont(Lap)
RajzolObjektumVéletlenPont(Lap)
RajzolObjektumVéletlenPont(Lap)
amíg nem PontAzonosság(vP1Lap; vP2Lap)
ElnevezPont(vP1Lap, A)
ElnevezPont(vP2Lap, B)
TörölObjektumVéletlenPont(Lap)
RajzolEgyenes(e; A, B)
Eljárás vége

Feladatok:

1.) Töröljük a *Lap* objektumról a *P* nevű pontot!

2.) Döntsük el az *A*-*B* pontpárról, hogy azonosak-e, és amennyiben igen töröljük az *A* pontot!

3.) Döntsük el egy *P* pontról, hogy rajta van-e egy *A* és *B* pontokkal adott egyenesen!

4.) Szerkesszünk *C* csúcsponttal szöget, melynek egyik szárán egy tetszőleges helyzetű, de *C*-től különböző *A* pont, másik szárán hasonló feltételekkel egy *B* pont helyezkedik el (jelölése: $ACB\angle$)!

5.) Döntsük el az $ACB\angle$ -ről, hogy derékszögű-e!

6.) Egy *A* és *B* pontokkal adott egyenesen vegyünk fel egy *C* pontot úgy, hogy *C* az *A* és *B* pontok közé essen!

7.) Szerkesszünk szakaszfelező merőlegest!

8.) Harmadoljunk egy *A* és *B* végpontokkal adott egyenes szakaszt!

9.) Döntsük el, hogy egy adott *P* pont egy adott *k* kör belsejében van-e!

10.) Szerkesszük meg adott szög szögfelező egyenesét!

11.) Szerkesszünk érintőt egy tetszőleges kör tetszőleges pontjában!

12.) Szerkesszük meg az *A*, *B*, *C* pontokkal adott háromszög súlypontját!

A feladatok megoldásához csak az automata által ismert hét eljárást használhatjuk fel! Az eldöntési feladatok esetében a „látszik” válasz nem releváns, hiszen rajzaink csak modellezik az absztrakt geometriai teret (maga a geometriai tér nem látható)!

Az automata eljárásainak megfelelő kombinálásával elemi szerkesztési algoritmusokat hozhatunk létre. Az összetett szerkesztést igénylő feladatok megoldásai sok esetben éppen ilyen elemi szerkesztési eljárásokból, esetleg azok többszöri ismétléséből állnak. A szükséges elemi szerkesztési eljárások algoritmusát ezért elegendő csak egyszer leírni, kódjukat „elkülöníteni”, majd egy egyedi névvel ellátni. Az eljárások minden további alkalmazásánál egyszerűen hivatkozzunk az eljárásra csupán nevével, kódja nélkül!

A szerkesztést igénylő feladatok megoldásának első nyers változatát az ismert elemi geometriai szerkesztések segítségével modellezzük, majd fejlesszük a megoldásokat az egyenes rajzoló példa megoldásánál mondottak figyelembe vételével!