LEGO® Education SPIKE™ Prime segédlet

LEGO® Education SPIKE™ 3.x.x verziószámú programhoz.

Algoritmus és vezérlő szerkezetek – alapok

szekvencia – elágazás – ciklus

Dokumentum verzió: v_01

Utolsó módosítás: 2026. május 17.

STEM Kuckó

Schlepp Péter 2025.

Bevezetés

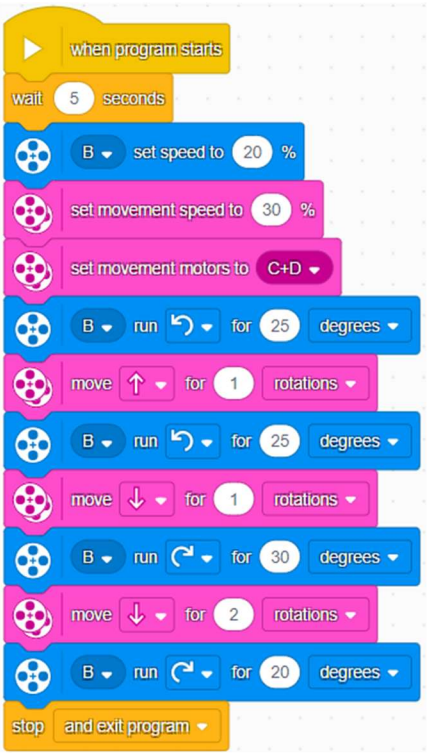
Mielőtt tovább lépnénk a LEGO® Education SPIKE™ Prime programozásában, röviden szót kell ejteni az algoritmusokról. Az algoritmus szó és fogalom a matematikából ered, de a számítástechnikai kultúra elterjedése ültette át a köznyelvbe.

> **Algoritmuson vagy eljáráson olyan megengedett – véges számú – lépésekből álló módszert, utasítás(sorozato)t, részletes útmutatást, receptet értünk, amely valamely felmerült probléma megoldására bizonyított és hatékony megoldást ad.**

Például eljárást, algoritmust, receptet lehet adni egy asztal vagy egyéb bútor összeszerelésére, valamilyen étel elkészítésének módjára, otthonról az iskoláig vezető út megtalálására, vagy éppen két egész szám összegének kiszámolására.

Ez gyakorlatban azt jelenti, hogy az elvégzendő feladatot kisebb, egymáshoz csak meghatározott módon kapcsolódó részfeladatokra bontjuk. A cél az, hogy a teljes feladat olyan kis feladatelemekre legyen felosztva, amelyek egymással nincsenek átfedésben, egymáshoz meghatározott logika szerint kapcsolódnak, és mindegyik megoldható az adott programnyelv saját elemi eszközeivel. A részfeladatokat megoldva és azokat egymáshoz illesztve megszületik a teljes feladat megoldása.

Erre példa a robot karjának programozása a gyűrűs elem felemelésére.



A teljes feladat: a robot emelje fel a gyűrűs elemet, tolasson hátra, tegye le és tolasson tovább.

Részfeladatok, amelyek programnyelv elemi blokkjaival megvalósítható:

1. program indítása
2. várakozás 5 másodpercig
3. karmozgató motor sebességének beállítás
4. robotmozgató motorok sebességének beállítása
5. robotmozgató motorok csatlakozó portjainak beállítása
6. emelő kar felemelése 25 fokot
7. előre mozgás 1 tengelyfordulást
8. emelő kar tovább emelése 30 fokkal
9. tolatás hátra 1 tengelyfordulást
10. emelő kar leeresztése 30 fokkal
11. tolatás hátra 2 tengelyfordulást
12. emelő kar további leengedése 20 fokkal
13. program vége

A fenti 13, szavakkal leírt részfeladat felsorolás ennek a programnak az algoritmusa.

Programnyelvtől függetlenül minden program három alapvető (vezérlő)szerkezetből áll. Ezek a **szekvencia**, az **elágazás** és a **ciklus**.

A szekvencia utasítások egymás utáni végrehajtása.

A bal oldalon látható program a következő algoritmus alapján készült:

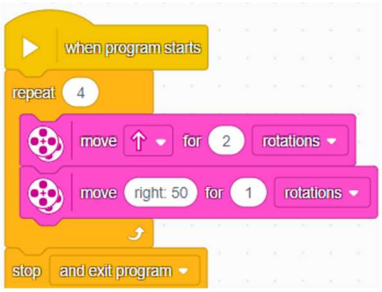
- menj előre
- fordulj jobbra
- menj előre
- fordulj jobbra
- menj előre
- fordulj jobbra
- menj előre
- fordulj jobbra

Értelmezve az algoritmust, ennek a programnak a végrehajtása során a robot egy négyzetet jár be. A program 8 utasítást tartalmaz, amelyeket egymás után hajtódnak végre. Ez a program egyetlen, 8 blokkból/utasításból álló szekvenciát tartalmaz.

Látható, hogy az utasítások sorrendje nem változtatható meg, csak a megadott sorrendben történő végrehajtás eredménye lesz a négyzet bejárás. Számtalan olyan feladat van, amelyre készített programban/algoritmusban néhány lépés/utasítás felcserélhető és ez a csere nem befolyásolja a végrehajtás eredményét. Ilyen esetekben az algoritmuskészítés előtt el kell döntenünk, hogy mi lesz az általunk követni kívánt végrehajtási sorrend. Az utasítások kötött sorrendje egyik eleme az algoritmus azon tulajdonságának, amit egyértelműségnek nevezünk. Az egyértelműségnek, mint tulajdonságnak eleme az is, hogy a leírás alapján bárki el tudja végezni a tevékenységet. Az egyértelműség mellett az algoritmus fontos jellemzője még az, hogy véges lépés után befejeződik.

Észrevehető. hogy a négyzet bejárás programban vannak ismétlődő utasítások. Ezek a menj előre – fordulj jobbra – utasítások. Ez az utasítás pár négyszer ismétlődik. Az ilyen ismétlődést nevezzük **ciklus**nak. A ciklus az algoritmus és így a program azon eleme, amely valamilyen feltételtől függően utasítás, programrész ismétlést valósít meg. A feltételt ciklusfeltételnek, az ismétlődően végrehajtandó utasításokat ciklusmagnak nevezzük.

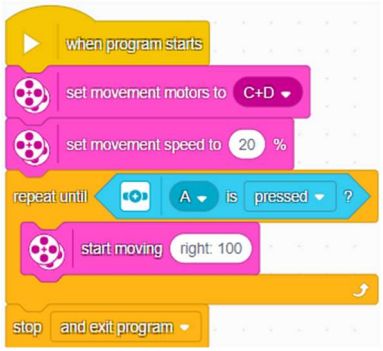
Ciklusokból háromfélét különböztetünk meg.



Az egyik a **számolt ciklus**. Ennél a ciklusnál a meghatározzuk, hogy a ciklusmag hányszor hajtódjon végre. Számolt ciklust a Controll blokkcsoport repeat blokkjával tudjuk megvalósítani. A négyzetbejárás esetében a ciklusmagnak– menj előre – fordulj jobbra  négyszer kell végrehajtódnia.

A ciklusok másik fajtája az **előltesztelős ciklus**, ahol a ciklusmag végrehajtása előtt történik a feltétel kiértékelése. Ha a feltétel igaz, akkor végrehajtódik a ciklusmag és újra megtörténik a feltétel kiértékelése. A ciklusmag mindaddig végrehajtódik, amíg a feltétel igaz. Amikor a feltétel hamis értéket ad vissza, kilép a ciklusból és a következő utasítás hajtódik végre. Ilyen ciklusnál előfordulhat, hogy a ciklusmag utasításai egyszer sem hajtódnak végre. A LEGO® Education SPIKE™ Prime programozási környezetben ebben az értelemben nincs előltesztelős ciklus. Ami leginkább hasonlít erre a ciklustípusra az a várakozás wait until.. blokk. Ahogy a Szenzorok használata – alapok (03_spike-v3_sensor_basic.pdf) segédletben is olvasható ez a blokk egy olyan paramétert – feltételt – vár, amely igaz vagy hamis érték lehet. Ameddig a feltétel hamis, a program nem lép tovább. Amikor a feltétel igaz értéket ad vissza, a program futása folytatódik.

A ciklusok harmadik fajtája a **hátultesztelős ciklus**. Ebben az esetben a ciklusmag utasításai egyszer végrehajtódnak, majd utána történik a ciklusfeltétel kiértékelése. A ciklusmag mindaddig végrehajtódik, amíg a feltétel hamis. Amikor a ciklusfeltétel igaz értéket ad vissza, a program kilép a ciklusból és a következő utasításra ugrik. A LEGO® Education SPIKE™ Prime programozási környezetben ezt a ciklus fajtát a Controll blokkcsoport repeat until blokkal tudjuk megvalósítani. A



példaprogramban a robot addig forog amíg nincs megnyomva az erő szenzor vagyis az is pressed értéke hamis. Amikor az erőszenzor megnyomására kerül, – az is pressed értéke igaz lesz, a program kilép a ciklusból és esetünkben a robot forgása és egyben a program is megáll. Ha a program indítása előtt benyomva tartjuk az erőszenzort, – az is pressed értéke igaz, akkor a forgás el sem indul.

FONTOS! Az előltesztelős és a hátultesztelős ciklusnál nem tudható, hogy a ciklusmag hányszor fog végrehajtódni.

A szekvencia és a ciklus mellett a harmadik vezérlési szerkezetet az elágazás. Az elágazásnál feltétel két ágra bontja a program végrehajtását. Ha a feltétel igaz, akkor az igaz ágban megadott utasítások, míg hamis érték esetén a hamis ágban megadott utasítások hajtódnak végre. Az elágazás speciális fajtái az egyágú elágazás, ahol valamelyik ág nem tartalmaz utasításokat, illetve a többágú elágazás. A LEGO® Education SPIKE™ Prime programozási környezetben a kétágú, illetve az egyágú elágazás megvalósítására találunk blokkokat. Többágú elágazás az egy- illetve a kétágú elágazás egymásba ágyazásával valósítható meg. Az egyágú elágazást a Controll blokkcsoport  if.. then blokkal, a kétágú elágazást  a if.. then .. else blokkal tudjuk megvalósítani. A kétágú elágazásra talán legjobb példa robotprogramozás témában a vonalkövetés, amelyet egy külön segédletben (Egyszerű vonalkövetés – 06_spike-v3_vonalkovetes.docx) dolgoztam fel.

Még néhány gondolat az algoritmusok kapcsán. Az algoritmus egyik fontos jellemzőjének definiáltuk az egyértelműséget. Az egyértelműség egyik eleme a világos, félreérthetetlen utasítások. Egy algoritmus megfogalmazásának a legtermészetesebb módja az lenne, hogy egyszerűen a saját szavainkkal elmondjuk a végrehajtandó tevékenységeket. Azonban a hétköznapi nyelv használatával a viszonylag egyszerű tevékenységeket is hosszasan „el kell magyaráznunk”! Ennek az oka az, hogy a hétköznapi nyelvben nincsenek kész kifejezéseink az algoritmusokban megszokott fordulatokra, így nem is alkalmas számítógépes algoritmusok elkészítésére. Az évek során kialakultak azok az eszközök és módszerek, amelyek nagy segítséget jelentenek abban, hogy az algoritmus készítés során megfeleljünk az egyértelműség kritériumának.

Ezek a módszerek a **hétköznapi nyelv**, a **folyamatábra**, a **struktogram** és a **pszeudokód**.

Ennek a segédletnek a keretein túlmutat, hogy részletesen ismertessem az algoritmus leíró eszközöket, viszont a továbblépéshez szükséges, hogy legalább egy módszert és annak alapjait ismerjük.

Pszeudokód:

A pszeudokód lényege, hogy az algoritmust mondatszerű elemekből építjük fel. Ez az algoritmus leíró módszer hasonlít legjobban a hétköznapi nyelvhez, azonban annyiban tér el tőle, hogy itt be kell tartanunk bizonyos szabályokat. A szabály az, hogy a struktúrák képzésére megállapodás szerinti formákat és szavakat használunk.

A továbbiakban egy-egy robotfeladat algoritmusának leírásához pszeudokódot használok!

A pszeudokód alapelemekei

Adatbekérés, adatkiírás

```
Be:...felsorolás...[megszorítások]
Ki:...felsorolás...[kiírási formák]
```

Szekvencia:

```
Tevékenység 1
Tevékenység 2
Tevékenység 3
```

Szelekciók:

Egyágú szelekció: Ha a Feltétel teljesül, akkor Tevékenység(ek) végrehajtásra kerül(nek), egyébként nem. A program az Elágazás vége után folytatódik. Itt az Elágazás vége el is hagyható.

```
Ha Feltétel igaz akkor
    Tevékenység(ek)
Elágazás vége
```

Kétágú szelekció: Ha a Feltétel teljesül, akkor Tevékenység(ek)1 kerül(nek) végrehajtásra, egyébként Tevékenység(ek)2. Mindkét esetben a program az Elágazás vége után folytatódik.

```
Ha Feltétel igaz akkor
    Tevékenység(ek)1
Egyébként
    Tevékenység(ek)2
Elágazás vége
```

Ciklusok:

Elöl tesztelő ciklus:

```
Ciklus amíg Feltétel igaz
    Tevékenység(ek)1
Ciklus vége
```

Hátul tesztelő ciklus:

```
Ciklus
    Tevékenység(ek)
amíg Feltétel igaz
```

Növekményes (számláló) ciklus:

```
Ciklusváltozó = ...tól ...-ig
    Tevékenység(ek)
Ciklus vége
```