DIGITÁLIS KULTÚRA

11.
PROGRAMOZÁS ALAPJAI SCRATCH

Összeállította: Kolman Krisztián

TARTALOMJEGYZÉK

A PROGRAMOZÁS ALAPJAI / ALGORITMIZÁLÁS ALAPJAI / SCRATCH FELÜLET HASZNÁLATA

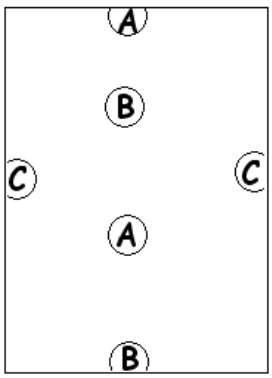
„A tudásod a legértékesebb vagyonod!"

A programozás tanulása során a legnehezebb feladat az, hogy az algoritmusokhoz és a programozáshoz kapcsolódó szaknyelvet, az elvont, általánosan megfogalmazott programozási feladatokat képes legyen megérteni valaki. A logikus gondolkodási képességet kell kialakítani, hiszen ennek hiányában nem lesz jó programozó valakiből.

1.) Logikai feladat:

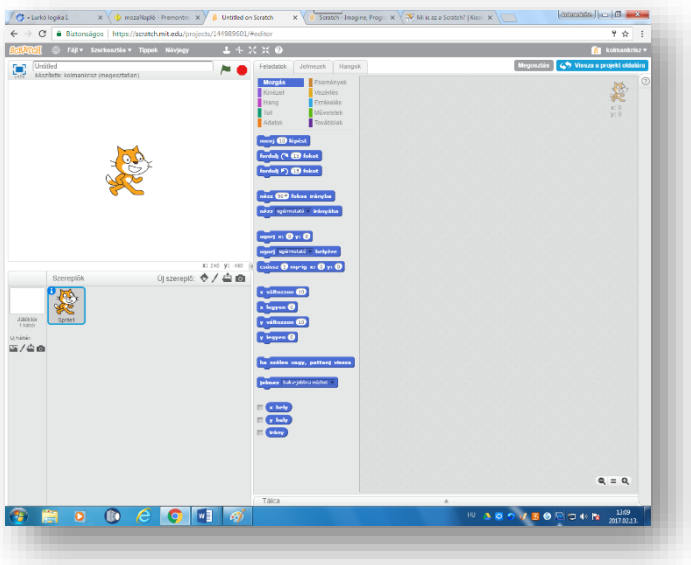
Kösd össze, ha tudod!

Próbáld meg összekötni A-t A-val, B-t B-vel, C-t C-vel három folytonos vonallal úgy, hogy a vonalak ne keresztezzék egymást, és ne menjenek le a papírról!



Mi is az a Scratch?

A Scratch készítői egy olyan környezet megírására vállalkoztak, amelyben a programozás módja és tárgya a programozással ismerkedők számára érdekes és látványos.



Miért könnyebb, és jobb ezzel a lehetőséggel megismerkedni, mint azonnal egy komoly programnyelvbe belevágni?

- A programokat, algoritmusokat **magyar nyelven** lehet megírni!
- A Scratch-ben való programozás **kirakójátékozásra** hasonlít: parancsokat és változókat felhasználva úgy lehet algoritmusokat összeépíteni, mint egy kirakót a darabjaiból. Ezek az elemek csak helyes módon illeszkednek egymáshoz.
- **Egyszerű logikával könnyen használható, és értelmezhető.**
- Az objektumorientált (szereplőközpontú) programozás támogatja az interaktív programok (például játékok) készítését, és segítségével kevésbé kell elvonatkoztatni a köznapi valóságtól.
- A környezet lehetőséget nyújt álló- és mozgóképi, hangi és zenei elemek vegyes használatára, így különböző tartalmakat és gondolatokat változatos médiaelemekkel lehet közvetíteni.
- Az elkészült programok feltölthetők a nemzetközi oldalra, ahol lehetőség van másoktól tanulni, mások munkáit értékelni, javaslatot tenni és fogadni.
- A magyar nyelv beállításához nem kell kiegészítőket telepíteni: a többi fordítással együtt beépítették a programba.

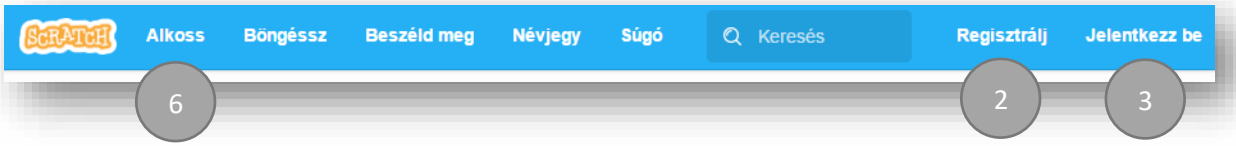
Hogyan álljunk neki? Melyek az első lépések?

Első dolgunk regisztrálni magunkat a felületen! Azért, hogy menthetőek, visszakereshetőek legyenek a munkáink, illetve lehetőség legyen a másokkal való megosztásra is!

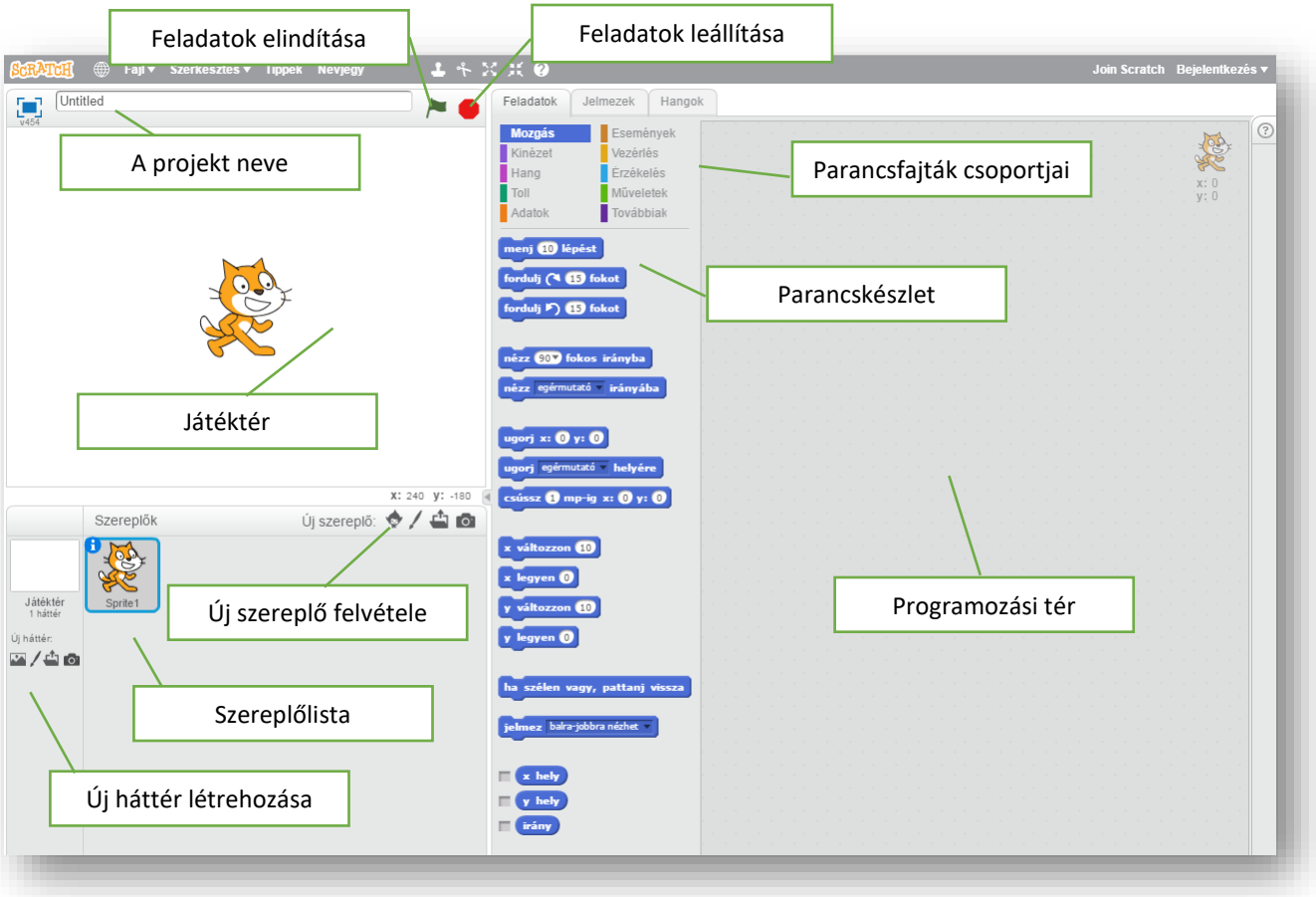
De lehetőség van az elkészített programokat lementeni a saját gépünkre, illetve a lementett programjainkat visszaolvasni gépünkről!

Tehát az első lépések:

1. A Google keresőbe beírni: „**scretch online”**, majd **https://scratch.mit.edu/** linkre lépni!
2. Az **online felületen regisztrálni** kell, majd a megadott email címre kapott levélben megerősíteni!
3. A regisztráció végeztével kezdhetjük a munkát. **Jelentkezz be**!
4. Megnézhetünk mások által készített programokat.
5. Nézz körül a felületen, bátran kattints rá bemutató videókra, programokra, leírásokra!
6. A fenti menüsávon az **„Alkoss” gomb**ra lépve megismerkedhetünk a programozói felülettel!



Melyek a kezdőképernyő részei? Milyen a Scratch felülete?



Hogyan tudom Scratch-et használni, ha nincsen net kapcsolatom?

Létezik offline verzió is. A következő linkről letölthető: https://scratch.mit.edu/scratch2download/

1. LECKE / AZ ELSŐ PROGRAM ELKÉSZÍTÉSE

2. Logikai feladat:

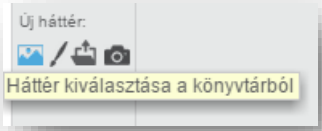
A nádasban néhány kígyó, béka és 2 gólya van. Összesen 9 fejet és 12 lábat számoltam meg. Hány kígyó, béka és gólya lakik a nádasban?

Megoldás:

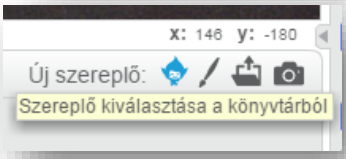
_____ db kígyó _____ db béka _____ db gólya

1.) Gyakorlat:

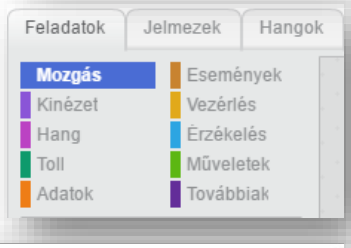
Először egy nagyon egyszerű, de viszonylag látványos programot készítünk. Az első programban egy kosárlabda pattan a földre, aztán a palánknak, majd újból a földre. A fiú pedig csak áll egy helyben, köszön, hogy „Helló”! Lépésről lépésre nézzük, hogy mit kell tennünk.



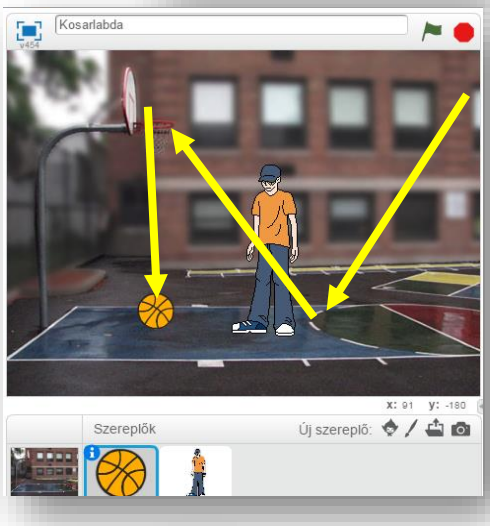
a.) Válasszuk ki háttérnek a kosárlabda pályát a könyvtárból! Tallózással kiválasztjuk (kékkel ki lesz jelölve) a „basketball-court1-b” képet és lent OK gomb megnyomásával beolvassuk!



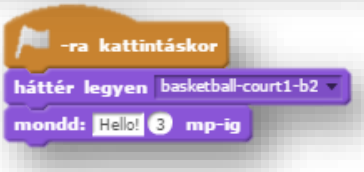
b.) A macskát a szereplők közül töröljük ki! Jobb egér a szereplőn, és törlés!
Két új szereplőt adunk hozzá! Kiválasztjuk (kékkel ki lesz jelölve) a „Basketball” és a „Breakdancer1” képeket, majd OK gombbal hozzáadjuk a listához!
Amikor programozzuk a szereplők cselekedeteit, mindig legyen kijelölve a szereplő!



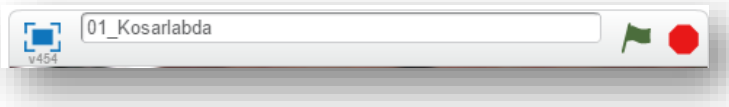
c.) Ennek a gyakorlatnak a megoldásában a „Parancsfajták csoportjai” közül csak a Mozgás; Kinézet; és az Események parancskészleteit használjuk. Az ezekben szereplő utasításokat alkalmazzuk!



- A programot úgy indítjuk, hogy kiválasztjuk a szereplőt, jelen esetben a fiút!
- A fiúnak a kép közepén kell állnia, és azt mondani, hogy „Helló”
- Ehez az utasításokat oda kell húzni a „Programozási tér”-re!
- Minden programot az „Események” parancskészletéből kiválasztva el kell indítani!
- Ebben az esetben a”⚑ –ra kattintáskor” parancsot válasszuk ki, és húzzuk a „Programozási térre”! Ezzel indítjuk a program futását!
- Alá behúzzuk a „Kinézet” parancskészletből a „háttér legyen …” parancsot, melyben a kosárlabdapályát állítom be.
- Ennek az egyszerű programnak az utolsó része a „mond: …Helo! __ mp-ig” parancs. Húzd az eddigiek alá!
- Ennél a parancsnál be lehet állítani, hogy mit „mondjon”, mit jelenítsen meg szövegként. Majd megadhatjuk, hogy hány másodpercig jelenjen meg a szöveg.



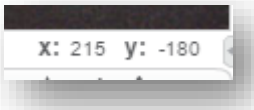
d.) Amikor végeztél, akkor a „Játéktér” felett adjuk meg a programunk nevét! Ebben az esetben a 01_Kosarlabda nevet adjuk meg!



e.) Végül futtathatjuk az egyszerű programunkat az -ra való kattintással! Ha lefutott a program, akkor lépjünk tovább!

f.) Válasszuk ki a kosárlabdát - jelöljük ki, aztán íjuk meg hozzá azt a programot, melynek az útját az előző oldalon lévő képen láthatjuk.

- Ezt a programot is a -ra való kattintással kezdjük! Húzd át a „Programozási térre”!
- Mivel a kosárlabda mérete irreálisan nagy, a méretét 50%-ra csökkentjük. A „Kinézet” parancskészletből a „méret legyen … %” parancsot választjuk, melyben a kosárlabda méretét állíthatjuk be.
- A „Mozgás” parancskészletből kiválasztjuk az „Ugorj x: ____ y: ____” parancsot! Ezzel megadjuk a labda kiindulási pontját!
- A következő három utasítással megadom, hogy milyen úton, és mennyi idő alatt, milyen utat járjon be a labda! Ehhez a „csússz __ mp-ig x: ___ y: ___” parancsot használom a „Mozgás” parancskészletből! Minden csúszás legyen 1 másodperc! A koordinátákat leolvashatjuk a játéktér jobb alsó sarkából! Beírhatjuk, hogy honnan hova menjen a labda!



g.) Ha összeszámolom, hogy a két programrész mennyi idő alatt fut le, akkor kiderül, hogy 3-3 mp alatt befejeződik! Futtassuk a -ra való kattintással!

h.) Az esetleges hibák javítása, többszörös futtatás után, mindig mentsük el munkánkat! Ezt fent, a „Fájl” menü legördítése után a „Letöltés saját gépre” menüpont kiválasztásával tehetjük meg!
Mindig rendszeresen mentsük munkáinkat!

A „ Betöltés saját gépről” menüponttal, előzőleg készített, vagy lementett programokat olvashatunk be!

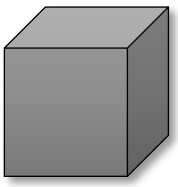
1. Önálló programozási feladat:

a) Az előzőek alapján, készíts egy 02_Kosarlabda nevű programot!
b) Használd háttérnek a „basketball-court1-b” képet!
c) A két szereplő „Dan” és „Champ99” legyen! A méretüket arányosan csökkentsd, vagy növeld, ha szükséges!
d) Az előző programban használt labdát ismételten alkalmazd!
e) A szereplőket a minta szerint helyezd el!
f) A feladat az, hogy „Dan” dobjon kosárra, „Champ” védekezzen!
g) A labda ívesen repüljön a kosárba, majd essen le a földre!
h) A kész programot mentsd a megadott helyre!

2. LECKE / AZ ELSŐ ANIMÁCIÓ ELKÉSZÍTÉSE

3. Logikai feladat:

Hány egyenlő nagyságú kockára van legalább szükség ahhoz, hogy egy újabbat hozzunk létre?



A válasz: _____ db

2.) Gyakorlat:

A következő programban elkészítjük az első animációnkat. Egy denevért fogunk reptetni faltól falig, oda-vissza.
Nem automatikusan, magától repül majd, hanem szóköz lenyomására hajt végre egy utasítássort, melyet ha folyamatosan nyomuk, akkor úgy néz ki, mintha repülne!

a.) Indítsunk egy új projektet! A neve legyen 03_Flying_bat!

b.) Válasszunk ki háttérnek egy egyszerű kék felületet! Legyen a „blue sky3” nevű!

c.) Töröljük ki a macskát a „Játéktérről”! Jobb egér a szereplőn, és törlés!

d.) Vegyük fel új szereplőnek a „Bat1”-et!

e.) Ennek a szereplőnek két „állása” van. Jobb oldalon látható, amikor a szárnya fent, illetve amikor a szárnya lent van.

f.) A program lefutásának indítása ebben az esetben szóközzel történik. Húzd a „Programozási tér”-re, az „Események” közül a 

parancsot!

g.) Majd állítsuk be, hogy 

!

h.) Azért, hogy a denevér oda-vissza tudjon repülni fel tudunk használni egy előre legyártott hasznos parancsot! Ez a 

parancs a „Mozgások” közül!

i.) Ha elért a széléhez, meg kell fordítani a denevért. Erre a parancs, a 

!

j.) A denevér előrehaladását a 

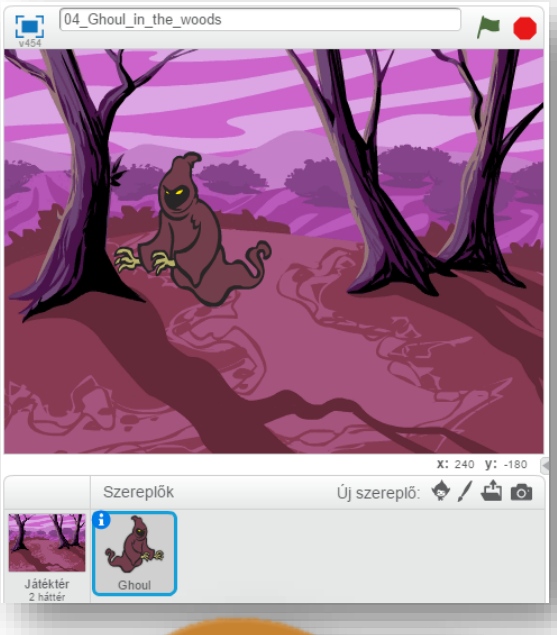
paranccsal biztosítjuk!

k.) Végül a jelmez „váltását” a 

, hogy a denevért „repülni” lássuk, ezzel a paranccsal biztosítjuk!

l.) A kész programot jobboldalon látjuk! A szóköz folyamatos nyomogatásával tudjuk futtatni programunkat, úgy, hogy a denevért repülni lássuk, és faltól falig oda-vissza haladjon!

3.) Gyakorlat:



A következő programban az előzőhöz hasonlóan egy szereplő közlekedik, szóközöket nyomva 10 lépésenként, de itt már nem egyenes vonalban, hanem a jobbra, és ballra, gombokkal 15 fokonként elforgatva az irányt! Tehát egy szellemnek kell „kísértenie” az erdőben!

a.) Indítsunk egy új projektet! A neve legyen 04_Ghoul_in_the_woods!

b.) Háttérnek tegyük be a képek közül a „Woods” nevűt!

c.) Szereplőnek vegyük fel a „Ghoul” nevűt! Ennek is két állása van, tehát mozgás animált lesz!

d.) Ehhez a programhoz három alprogramra lesz szükség. Ezeket a szereplőhöz tartozó „Programozási tér”-hez kell húzni az utasításokat!

e.) Az alap program szinte teljesen ugyan az mit az előző programban, amikor egyenesen haladt a szereplő!

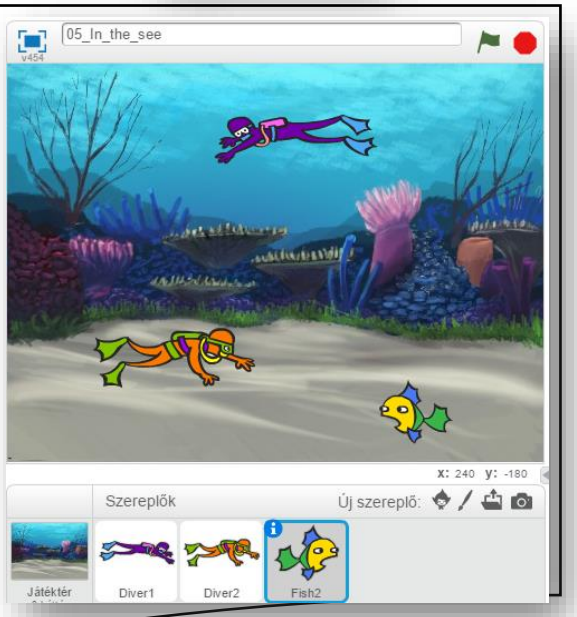
f.) Majd két alprogramot kell készíteni, ahhoz hogy irányt tudjon váltani a szereplő!

- Az „Események” parancskészletéből kiválasztjuk azt, amelynél valamely gomb lenyomásakor futtatja a programrészt! Az egyiknél a bal gomb lenyomásakor a szereplő elfordul 15 fokot, és vált a következő jelmezre!
- Majd megismétlem jobb gombbal a másik irányba is!
- Az alprogramot jobb egérrel, lehet duplikálni!

g.) Végül teszteld a kész programot!

h.) Mentsd le a saját gépedre a megadott néven, hogy visszakereshető legyen!

2.) Önálló programozási feladat:



a.) Indítsunk egy új projektet! A neve legyen 05_In_the_see!

b.) Háttérnek válaszd ki az „Underwater2” képet!

c.) Vegyél fel három szereplőt! A „Diver1”, „Diver2”, és a „Fish2”-t!

d.) A programban búvárok, és egy hal ússzon a tengerben, az előző programokban leírtak szerint, különböző irányokban!

e.) A programrészeket nem kell mindig újra készíteni. A programrészeket rá lehet húzni más szereplőkre, és így átkerül a másik szereplő Programozási terére!

3. LECKE / A HÁTTÉR VÁLTOZTATÁSA / A SZEREPLŐK VÁLTOZTATÁSA / HANGOK ALKALMAZÁSA

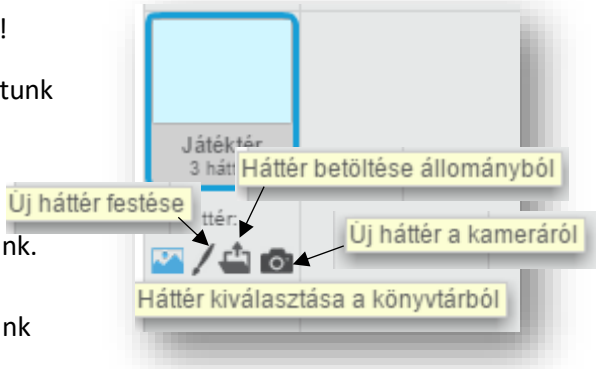
4. Logikai feladat:

Egy gyufaszál áthelyezésével tedd igazzá az egyenlőséget!

(Satírozd ki az eredeti helyéről, rajzold be az új helyére!)

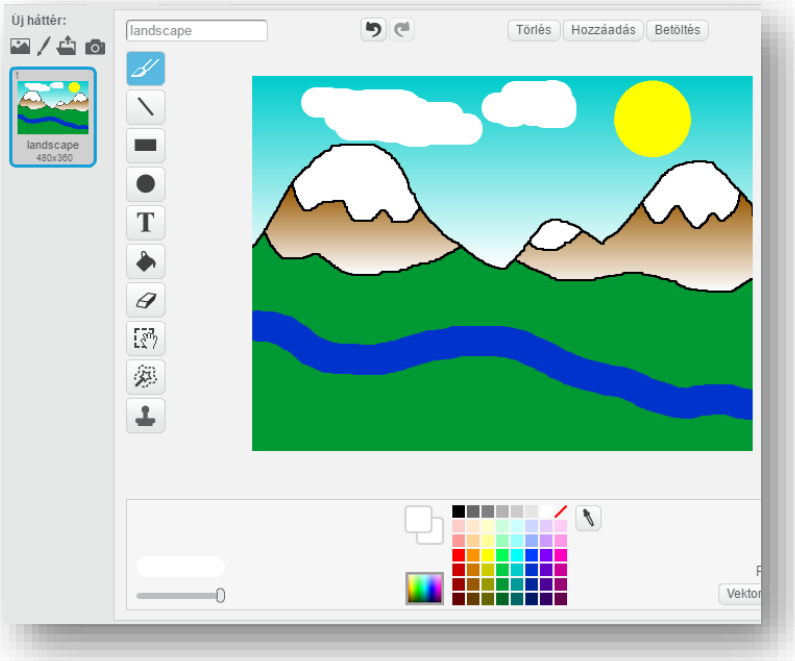
A háttér mindig nagyon fontos része egy programnak. AScratch-ben több lehetőség is van, hogy a hátteret beállítsuk, kiválasszunk!

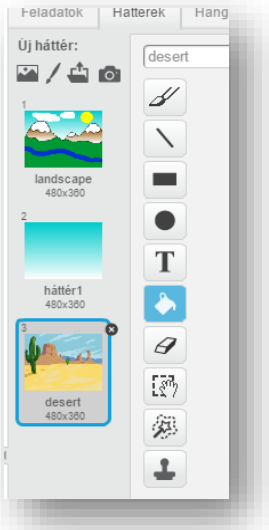
- Már megszokott módon a saját tárhelyről beolvashatunk meglévő képet!
- Új hátteret készíthetünk, saját ötletek, szükségletek szerint. A Paint programhoz hasonlóan. Vektoros és pixeles képeket is készíthetünk.
- Saját gépünkről tölthetünk be képeket, tallózással!
- Végül saját számítógépünk kamerájával is készíthetünk képeket, ha engedélyt adunk a programnak, hogy hozzáférhessen a kamerához!



4.) Gyakorlat:

- Készítsük el a jobb oldalon látható „Landscape” nevű képet!
- A képet pixelesen rajzoljuk!
- Először válasszuk ki az ecset eszközt, és fekete színnel rajzoljuk meg a hegyek sziluettjét! Figyeljünk arra, hogy zárt legyen az alakzat a lap szélétől a lap széléig!
- Majd rajzoljuk meg a hóhatárt és az erdőhatárt!
- Aztán Kitöltésnél válasszuk ki a világoskék színt, és állítsuk színátmenetesre! Kattintsunk a kép felső részére!

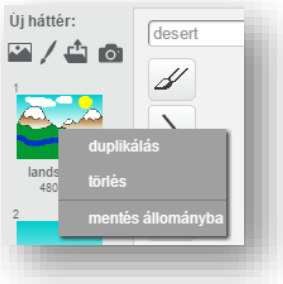




- Ugyanígy legyen színátmenetes barna a hóhatár alatti rész is!
- Rajzoljuk meg a folyót egyszínű sötétkék ecsettel, de a vonal vastagságát állítsd be!
- Majd színezzük a maradék területet egyszínű zöldre, az ecset eszközzel!
- Készítsünk napot az ellipszis eszközzel, a Shift gomb lenyomásával! Állítsuk kitöltöttre az alakzatot!
- Aztán vastagra állított fehér színű ecsettel rajzoljunk felhőket a minta alapján!
- Ehhez a programhoz hozzunk létre még egy hátteret „hattér1” néven, melyen csak egy kékből fehérbe átmenő kitöltése legyen!
- Végül a beépített képek közül szúrd be a „desert1” nevűt!

Egy programhoz több hátteret is felhasználhatunk. Az előző oldali képen látható, hogy hogyan rendeződik listába! A programokon belül a parancsoknál van lehetőség a nevek alapján felhasználni őket!

Ha olyan programot készítünk, ahol két, szinte ugyanolyan képre van szükségünk, akkor lehet duplikálni a képet, majd az egyiken elvégezzük a változtatásokat! A miniatűrön jobb egér és duplikálás!



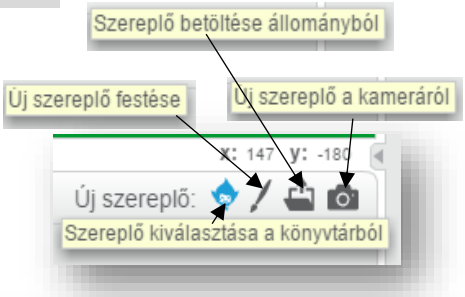
Ha el szeretnénk menteni az elkészített háttérképet, mert máskor is fel akarjuk használni, akkor a miniatűrön jobb egeret kell nyomni, és a „mentés állományba” almenüt kell kiválasztanunk! Tallózással megadni, hogy hova szeretnénk elmenteni!

A gyakorlat végén mentsd a projektet 06_bg_fig_voi néven!
(bg=background=háttér; fig=figure= alak, szereplő; voi=voice=hang)

5.) Gyakorlat:

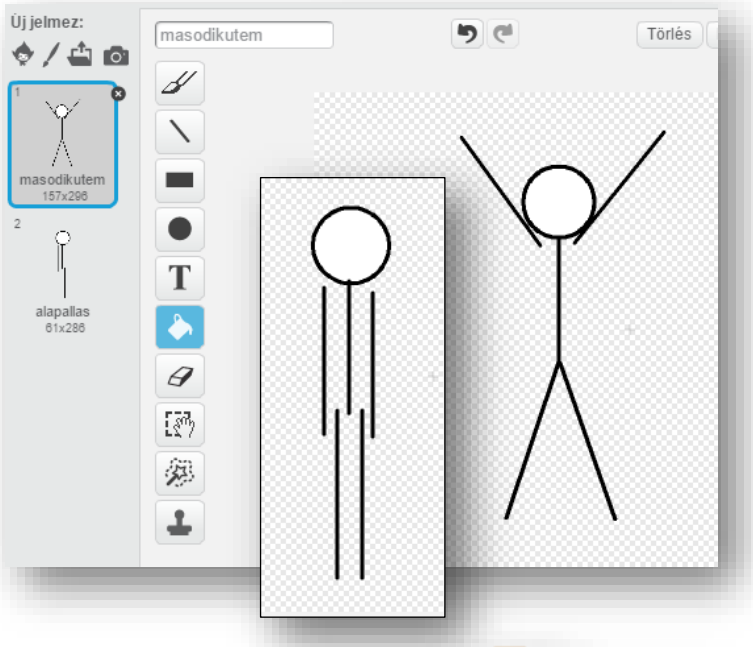
A programok legfontosabb résztvevői a szereplők.

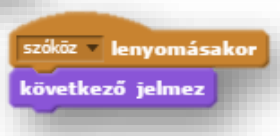
Négyféleképpen olvashatsz be, illetve hozhatsz létre szereplőket.



Ebben a feladatban egy kétütemű gyakorlat bemutatásához szükséges kétállású „pálcikaembert” rajzolunk!

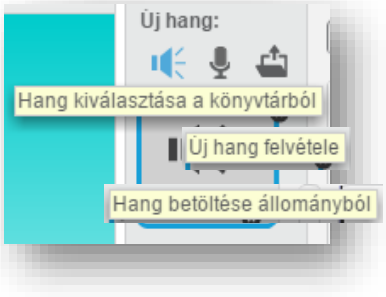
- Az előzőleg mentett projektben dolgozzunk tovább.
- Válasszuk ki az „Új szereplő festése” ikont!
- Az első rajz neve legyen „masodikutem”! Vékony vonalakból és egy körből rajzoljuk meg a képen látható figurát! A kört töltsük ki fehér színnel!
- A második rajz neve legyen alapállás! Ugyanabban a pozícióban, ahol a másikat megrajzoltad, készítsük el a mintán látható, álló figurát!
- Az új szereplő neve legyen: Gimnasztika01! Mentsünk rá a 06_bg_fig_voi fájlra!
- Ellenőrizzük le a munkánkat úgy, hogy a programozási térre húzzunk át két egyszerű parancsot! A jobboldali képen látható utasításokkal, szóközök nyomásával megnézhetjük, hogy jól dolgoztunk-e?





6.) Gyakorlat:

Hangefektetket is adhatunk a programunkhoz! Kiválaszthatunk meglévő hangokat könyvtárból vagy saját forrásból!

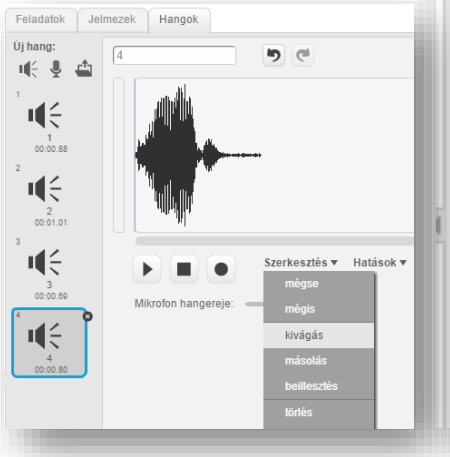


- Vegyünk fel szellem hangot a számítógépünk mikrofonján keresztül!
- Majd ha szükséges vágjuk meg, szerkesszük a lehetőségek szerint!
- A hang neve legyen „szellem_hang”! Mentsük a 06_bg_fig_voi-t!

3.) Önálló programozási feladat:

Készíts egy új programot, melyben egy tornaszobában, egy pálcikaember négyütemű fekvőtámaszokat végez folyamatosan!

A program úgy fusson le, hogy közben az ütemek számát halljuk a hangszórón keresztül!



a) Hozzál létre egy új projektet, melynek neve legyen 07_gimnastic!

b) Készítsél egy új hátteret! Rajzoljál a mintán látható tornaszobához hasonlót! A háttér neve legyen: tornaszoba!

c) Hozzál létre egy új szereplőt negyutemu néven!

d) Váltsál vektoros képalkotásra, és rajzold meg a három különböző ütemet a mintán látható minta szerint! A guggolást csak duplikálni kell a megfelelő helyre!
A képeket kicsit elforgatva rajzold meg!
Nevezz meg minden ütemet szám szerint!

e) Vegyél fel új hangokat a számítógéped mikrofonján keresztül! (egy; kettő; három; négy) Ha szükséges vágd le a felesleges részeket!

f) Készítsd el a programot! Húzd a megfelelő parancsokat a programozási térre a megfelelő sorrendben!

g) Futtasd, teszteld a kész programot!

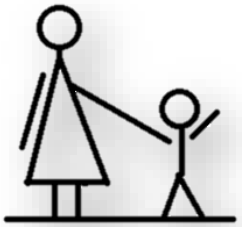
h) Végül mentsd (exportáld) 07_gimnastic néven a saját gépedre!

4. LECKE / PARANCSOK MEGISMERÉSE (MOZGÁS, KINÉZET, ESEMÉNYEK)

5. Logikai feladat:

Orsi az öccsével sétál. Megkérdezi egy ismerős, hány évesek. Két és félszer annyi idős vagyok, mint az öcsém, tavaly háromszor annyi voltam, tavalyelőtt pedig négyszer annyi, mint ő. Hány évesek most a gyerekek?

Megoldás: _____

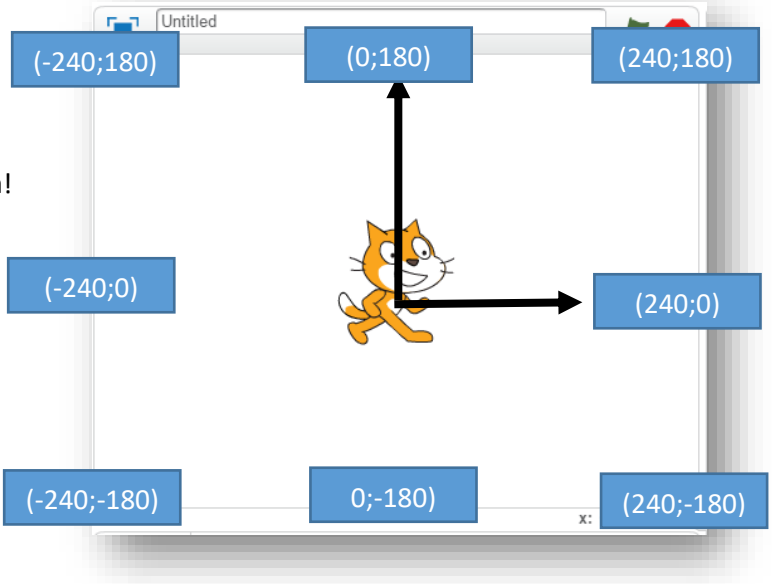


Hogyan mozgunk a „Játéktéren”?

A „Mozgás” parancskészlet utasításai egy koordinátarendszer X és Y pontjai alapján történik, a szereplő alapállapotához viszonyítva!

A „Játéktér” 480*360 képpont méretű!

A számozás a lap középpontból (0,0) indul, úgy hogy a bal felső saroktól (-240;180) –tól; a jobb alsó sarokig (240,-180)-ig tart.



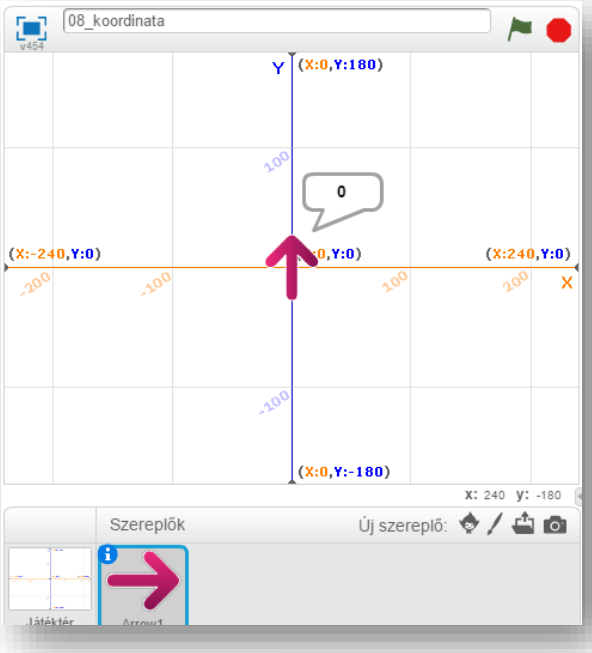
7.) Gyakorlat:

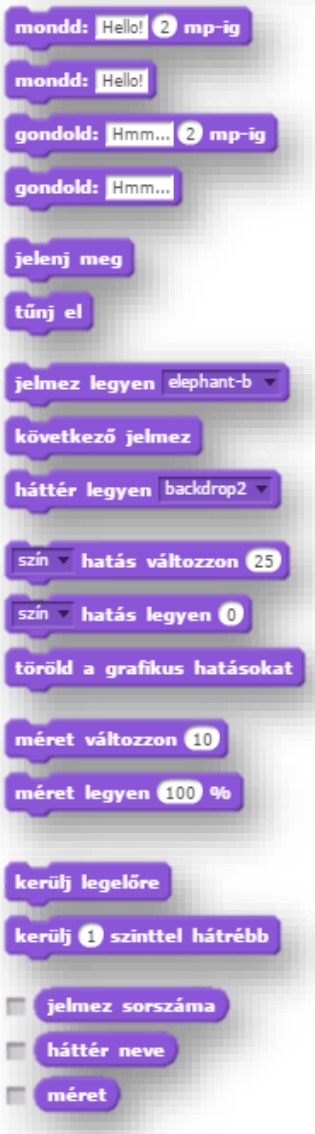
A feladat: a játéktéren egy nyilat mozgatni a kurzormozgató nyilakkal!

- A projekt neve legyen: 08_koordinata!
- A háttér legyen: „xy-grid kép”!
- A szereplő legyen: „arrow1”!
- A zászlóra kattintva induljon a program!
- A program szereplője (0,0) koordináta pontból induljon, és mindig a 0 fokos irányból!
- A kurzormozgató gombokkal 10 képpontot haladjon a szereplő, és mindig álljon abba az irányba amerre megyünk! Ezt a szereplő forgatásával érjük el, és ne „következő jelmez”-zel!
- Ha a széléhez ért a szereplő, mindig pattanjon vissza!
- Gombnyomáskor mindig „mondja”, írja ki egy buborékba, hogy az adott tengelyen (irányváltott) éppen mennyi a koordináta képpontja!
- Szóköz lenyomásakor mindig térjen vissza a kiindulópontba a (0,0) origóba, és vegye fel a kiinduló helyzetet!

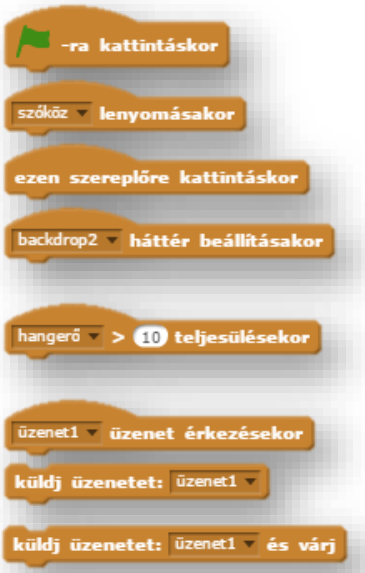
(A megoldás a következő oldalon! Ne fordíts, amíg nem próbáltad megoldani!)

Az előző feladat megoldása:





- A „Kinézet” parancskészletben a szereplők buborékban mondhatnak, illetve gondolhatnak valamit.
- Megjelentethetjük, eltüntethetjük a szereplőket, válthatunk jelmezeket.
- Megváltoztathatjuk a mértét, képpontban és százalékban.
- A szereplők rétegenkénti sorrendjét adhatjuk meg.
- A szereplők bizonyos tulajdonságának változását érhetjük el! (pl.: szín)
- Az „Események” parancskészletben a programok, programrészek indítását állíthatjuk be!



4.) Önálló programozási feladat:

a.) Hozzál létre egy 09_goal nevű programot!

b.) A háttér legyen: goal1

c.) A szereplő legyen: Hanahh

d.) A mérete legyen 20%-al kisebb!

e.) Rajzolj egy labdát!

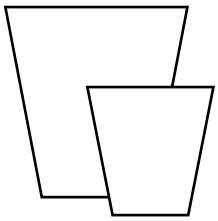
f.) A feladat az, hogy ha rákattintasz a Hanahh1-es állására, akkor váltson a Hanahh2-re, és a labda repüljön be a jobb felső sarokba, majd essen le a földre!

g.) A szereplő kiabálja, azt hogy „GóóóL!”!

5. LECKE / ISMÉTLŐDÉSEK, CIKLUSOK

6.) Logikai feladat:

Az iskola büféjébe OKÉÉ! pudingot szállítottak. Vaníliásból háromszor annyit, mint csokisból. Összesen 96 dobozzal. Mennyi érkezett az egyik, ill. a másik fajtából?



Megoldás: Vaníliás: _____ db Csokis: _____ db

Az eddigi programokat úgy írtuk meg, hogy csak akkor csináljon valamit, ha leütöttünk egy billentyűt. Ehhez a „Vezérlés” parancsok közül választunk!

Az ismétlődéseket és ciklusokat akkor használjuk, ha többször szeretnénk ugyanazt a műveletet lefuttatni.

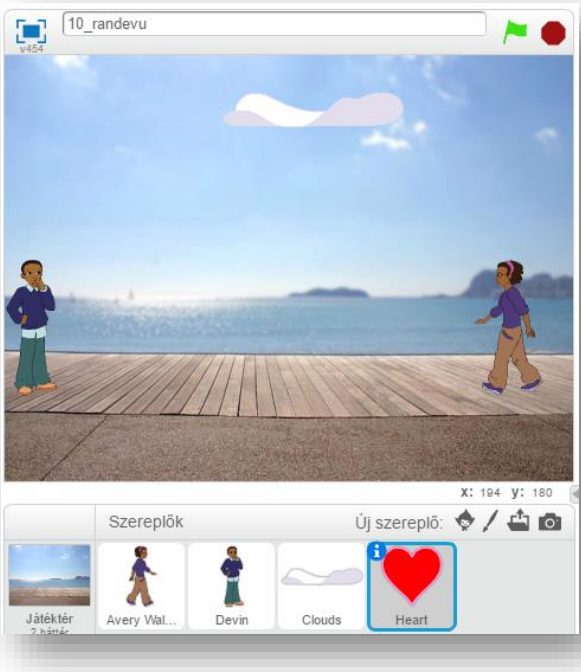
Ebben a leckében két ciklusfajtát fogunk megismerni:

mindig: folyamatosan ismétli a benne leírt parancsokat

ismételd: a megadott számú alkalommal ismétli a benne elhelyezett parancsokat

8.) Gyakorlat:

Ebben a programban egy fiú várja a lányt randevúra! A lány megérkezik, és oda megy hozzá. Majd elkezdenek beszélgetni, köszönnek egymásnak, és a fiú elhívja moziba! Eközben egy gomolyfelhő folyamatosan megy fent az égen! Amikor vége a beszélgetésnek jelenjen meg egy szív a páron, és növekedjen!

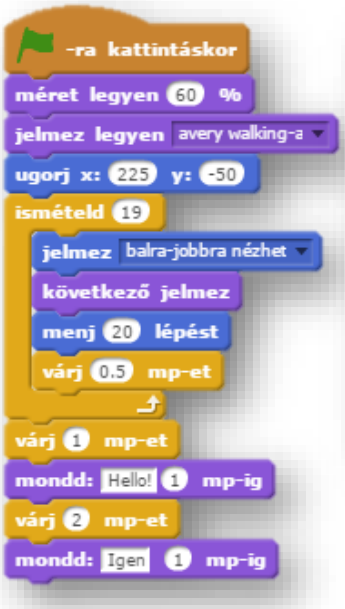


- A projekt neve 10_randevu legyen!
- A háttér legyen „boardwalk” kép!
- Vedd fel a négy szereplőt: „avery walking”; „Devin”; „Clouds”; „Heart”!
- Mind a négy programrész zászlóra kattintással induljon!
- A lány eredeti méretének legyen a 60 %!
- A lány álló helyzetből induljon! (avery walking-a)!
- A stég végéről induljon a fiú felé! Csak annyit haladjon, hogy megálljon a fiú előtt! 20 képpontnyit haladjon 0,5 mp-enként!
- Ez idő alatt a fiú, szintén elkezdi algoritmusát! Álljon a stég másik végén 60% méretben! Helyezd pontosan egy vonalba a lánnyal!
- Amikor a lány odaért a fiú elé, elkezdik a párbeszédüket! A fiú köszön 1 mp-ig, hogy „Szia!”! Majd a lány köszön 1 mp-ig „Helló!”! Aztán a fiú megkérdezi, hogy: „Eljössz velem moziba?”! Erre a lány „Igen”-t mond! Ezt ki kell számolnod, hogy mikor mennyit kell várnia a másiknak!
- A teljes program alatt fent az égen a felhő 0,25 mp-es késleltetéssel oda-vissza megy! A felhő (-10;130) –as koordinátáról induljon!
- Mivel a „Sziv”-re csak a beszélgetés végén van szükség, ezért az elején el kell tüntetni 15 mp-re!
- Ez az alakzat 0,1 pm-enként növekedjen 10 képponttal! Majd a végén tűnjön el a szív!
- A program írása közben, és a végén teszteld, javítsd az esetleges hibákat! Ments mindig!

(A megoldás a következő oldalon! Ne fordíts, amíg nem próbáltad megoldani!)

Az előző feladat megoldása:

Avery

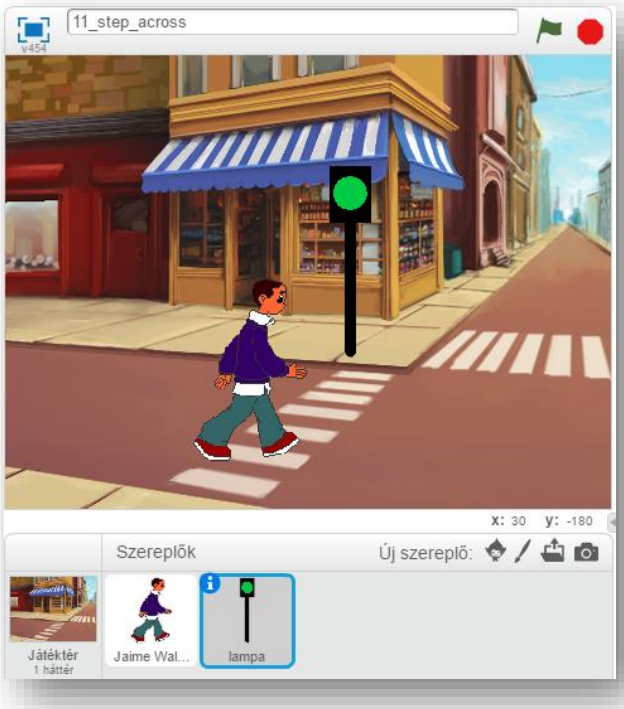


Devin

Clouds

Heart

5.) Önálló programozási feladat:



Ebben a programban egy útkereszteződésben egy fiú kel át az úttesten!

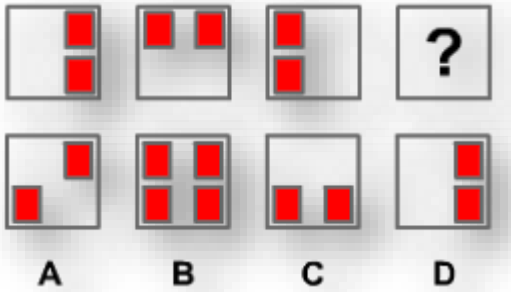
Úgy induljon a program, hogy a fiú áll a kép bal oldalán lent az út szélén, és vár, hogy a lámpa zöld legyen! Amikor átvált a lámpa, akkor a fiú átindul az úttesten! Amikor átért, menjen tovább a másik zebrán!

- A program neve legyen 11_step_across!
- A háttér legyen Urban2!
- A fiú legyen Jaime Walking szereplő!
- Rajzol egy jelzőlámpát és helyezd a mintán látható helyre! Legyen a neve „lampa”!
- Két állása legyen! Az egyik piros,a másik zöld! A neve is ez legyen!
- A fiú mérete legyen 100%!
- A rajzolt lámpa miatt, a fiú kerüljön előre!
- Akkor induljon el a fiú, amikor a lámpa pirosról zöldre váltott! Ezért az elején késleltesd az indulást 2 mp-et!
- A jelmezek váltásával várjál 0,5 mp-et!
- A fiú átlósan haladjon, mintha keresztbe menne át az úton! Ezért az x változzon 18 képpontot, az y változzon 10 képpontot! Így ár át az úton!
- Közben a fiú mérete csökkenjen -5 képpontot lépésenként!
- Amikor átért, akkor menjen tovább a másik zebrára, és a lámpa váltson pirosra!
- A program írása közben, és a végén teszteld, javítsd az esetleges hibákat! Ments mindig!

6. LECKE / A „HA - ____ - AKKOR” FELTÉTEL HASZNÁLATA

7.) Logikai feladat:

Figyeljük meg az ábrák rajzolatának változásait és próbáljuk meghatározni, hogy az alsó sorból milyen alakzat kerülhet a kérdőjel helyére.



A megoldás: _____ Miért? _____

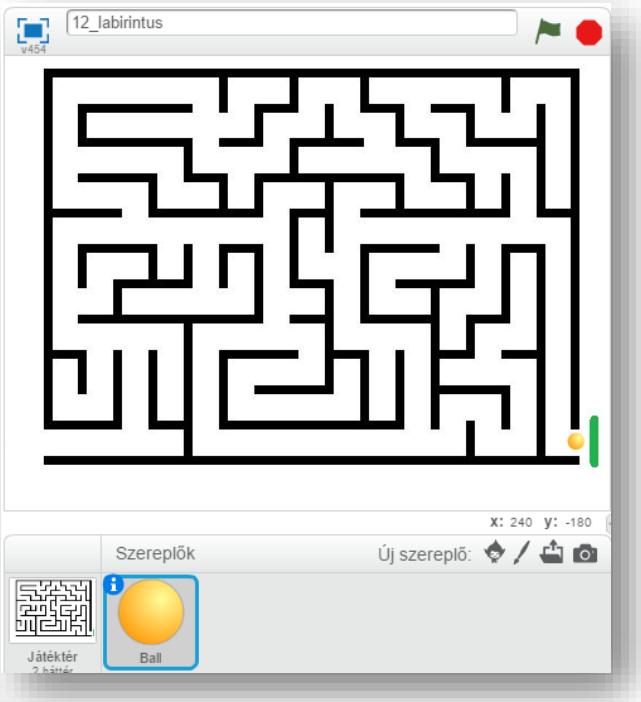
Amikor programokat készítünk, akkor a szereplők nem mindig ugyanúgy viselkednek helyzetekben. Ahhoz hogy változtatni tudjuk, feltételeket fogunk megadni. Ezért a „ha ____ akkor” feltételt fogjuk használni. A feltételben a „hatszögbe” be kell húznia parancsok közül, a hozzá hasonló hatszögű vizsgálatot!

9.) Gyakorlat:

A feladatban egy sárga kis labdát kell végigvezetni egy labirintuson úgy, hogy nem ér hozzá a falhoz, mert különben visszaugrik a kezdőpontra!

Amikor kiér a labda a labirintusból, és megérinti a zöld vonalat, írja ki egy szöveget!”

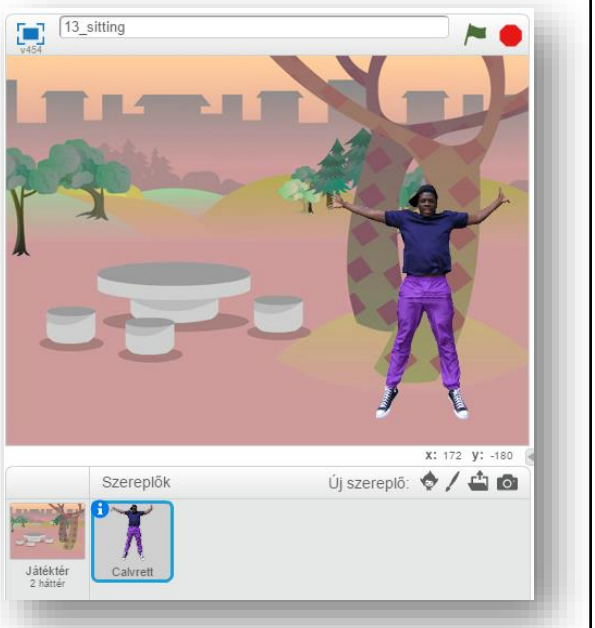
- A program neve legyen 12_labitrintus!
- A háttér legyen beolvasva mappából! A kép neve kep_labirintus.jpg!

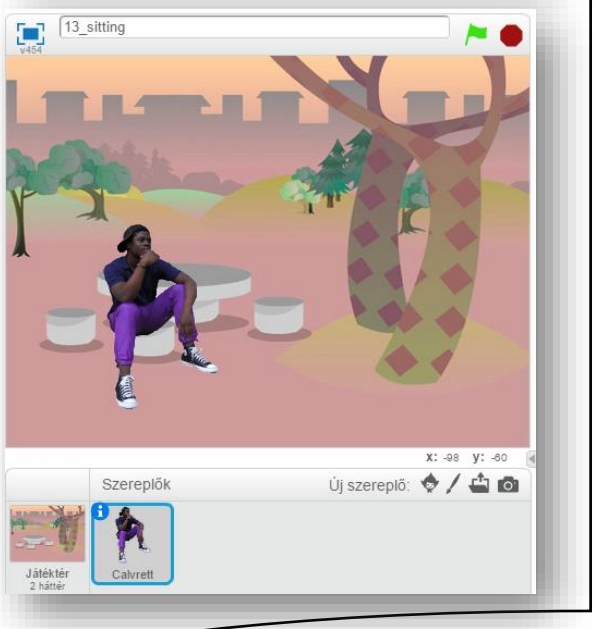


- Szereplőnek a „Ball” képet töltsük be!
- A labda méretét csökkentsük 30%-ra!
- A labda bal oldali bejárattól induljon! Koordinátái (-220;-125)!
- A kurzormozgató nyilakkal léptessük a labdát, koordinátánként 5 kp-tal!
- Ha a labda fekete színhez ér, akkor ugorjon vissza a kiinduló pontra!
- Ha a labda beért a célba, és hozzáért a zöld sávhoz, akkor írja ki két másodpercig, hogy „Nyertél!”
- A program írása közben, és a végén teszteld, javítsd az esetleges hibákat!

6.) Önálló feladat:

Ebben a programban a szereplő áll egy parkban, majd ha kattintok az egérrel a képen lévő székre, vagy asztalara, akkor 2 másodpercig leül rá, majd visszaugrik a kiinduló helyére!





- A projekt neve legyen 13_sitting!
- A háttér legyen: woods and banch!
- A szereplő legyen: Calvert!
- A program zászlóra kattintással induljon!
- A szereplő álló (ugró) állása legyen a kiinduló állapot!
- A jobb oldalon helyezkedjen el induláskor!
- Ha odakattintok valamelyik székre, vagy az asztalra, akkor ugorjon abba a pozícióba és változtassa meg az állását ülő pozícióba!
- 2 másodperc után menjen vissza a kiinduló helyzetbe!
- Ahhoz hogy ez működjön a „Vezérlés” parancsok közül szükség lesz a „mindig” utasításra, hogy figyelje a kattintást! Ne csak egyszer fusson le!
- A program írása közben, és a végén teszteld, javítsd az esetleges hibákat!
  Ments mindig!

Ismertető, összegző:

Az utasítások egymásba illeszthetők, mint a legók!

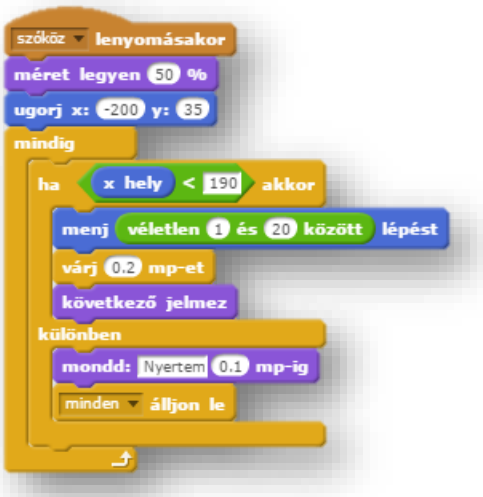
A fehér kör, illetve kapszula alakú mezőkbe értékeket írhatunk!

Tizedes számoknál pontot használunk! (pl.: 0.25)

A fehér kör, illetve kapszula alakú mezőkbe húzhatunk lekerekített szélű egyéb parancsokat, melyek értékeket adnak vissza! (pl.: „x hely”)

A hatszög alakú mezőkbe olyan parancsokat húzhatunk, amelyek „igen” vagy „nem” válasszal térnek vissza! Vagy az „Érzékelés” csoportból az oda illő hatszögű parancsokat húzhatjuk be! (pl.: szín érzékelése)

A legördülő mezőkből értelemszerűen választunk!

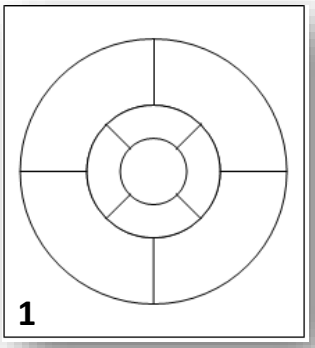


7. LECKE / A „HA - ____ - AKKOR; KÜLÖNBEN” FELTÉTEL HASZNÁLATA

8.) Logikai feladat:

Írjál 1-4 ig számokat úgy a tartományokba, hogy a szomszédos részben nem lehet két egyforma szám!

A külső részben van az 1-es szám, tehát nem lehet egyes a külső körgyűrűben!

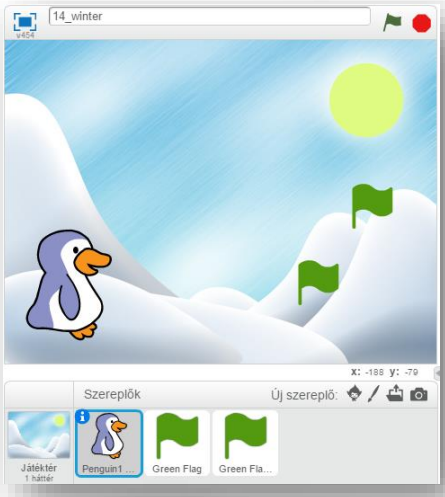


Az előző leckében tanult „Ha, akkor” feltételt kibővítve, egy „különben” lehetőséggel, újabb út nyílik a problémák megoldására. Kétfajta kimenetei lehetőséget biztosít. Itt is meg kell keresni a megfelelő vizsgálatot, és behúzni a „hatszögbe”.

10.)Gyakorlat:

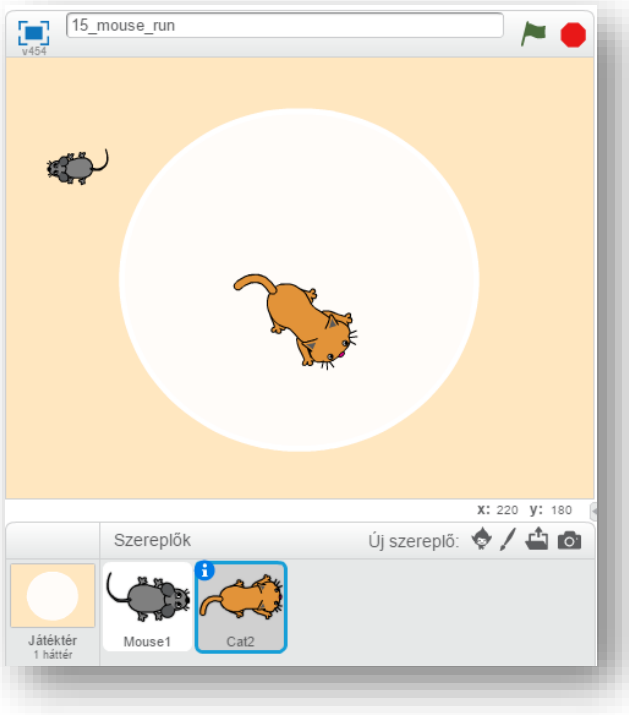
Ebben a feladatban egy pingvinnek kell felmennie a hegyre, és ha felért a hegycsúcsra, akkor gondolja magában, hogy: „Felértem!:)”! Használjuk az ebben a leckében megemlített „Ha, akkor, különben” vezérlő parancsot! Ha kell többször is!

Takard le a megoldást, hogy csak az utasítások alapján haladj!



- A projekt neve legyen: 14_winter!
- A háttér legyen: „slopes”!
- A főszereplő legyen a ”penguin 1”!
- A másik szereplő legyen egy zöld zászló! Amit duplikálni kell, mert kettő kell belőle!
- Ezeket a zászlókat a hegy oldalában helyezzük el! (Később kiderül majd, hogy miért!)
- A program szóköz lenyomására induljon!
- Adjunk meg egy kiindulópontot a szereplőnek a bal oldalon!
- A pingvin indulás után menjen egyenes vonalban előre, amíg el nem éri a zöld zászlót! Ha elérte, akkor menjen föl a hegyre!
- Az előzőt úgy érjük el, hogy vizsgáljuk azt, hogy elérte-e a zöld színt, vagy nem!
- Amíg nem érte el a zöld színt, addig az „x” változzon 10 kp-ot, az „y” 0 kp –t lépésenként!
- Amikor elérte a zöldet (zászlót) akkor az „x” 5 kp-ot, az „y” is 5 kp-ot változzon! Így megy fel a hegyre!
- A lépések közben a pingvin ugorjon a következő jelmezre!
- Ha eléri a nap sárga színét, akkor gondolja magában, hogy: „Felértem!:)”; és álljon le a program!
- A programozásnál kicsit fordítva kell gondolkodni! A program belsejéből kell kifelé haladni!
- Teszteld, és mentsd a megadott helyre!

7.) Önálló feladat:



Készítsd el a feladatot a következő instrukciók figyelembevételével!

A programban egy kisegér szaladjon a rózsaszín felületen! Ha a széléhez, vagy a fehér részéhez ér, akkor pattanjon vissza általad választott szögben, és forduljon meg! Az egér haladásának sebességét magad válaszd meg, de ne legyen se túl lassú, se túl gyors!

A macska a fehér körben ugyanazokat tegye, csak ha a rózsaszín részhez ér, akkor pattanjon vissza bizonyos szögben a kör széléről!

- A programot a „Ha, akkor, különben” vezérlőparancs felhasználásával készítsd el!
- A program neve legyen: 15_mouse_run!
- A programot szabadon választott módon indítsd el!
- Az egyik szereplőnek válaszd a mouse 1 figurát!
- Az egér eredeti mérete 50%-a legyen, azért, hogy átférjen a kör alatt és felett! A futás alatt a másik oldalra át tudjon menni!
- Az egérnek két állasa van, ezért minden lépésnél váltson jelmezt!
- A kiinduló pozíció mindig ott legyen, ahol az előző futást befejezte!
- Az irányokban és futási időben szabadkezet kapsz!
- Viszont egy bizonyos lefutás után automatikusan álljon le!
- A másik szereplőnek a „cat2”-t válaszd!
- A futáskor a macska 80%-os méretben fusson!
- A fehér körben ugyanolyan mozgásokat végezzen, mint az egér!
- A macska ugyanakkor álljon le, amikor az egér!
- Minden más felmerülő kérdésben szabadkezet kapsz!
- A programot írás közben többször teszteld!
- A végén mentsd a megadott helyre, a megadott néven!

8. LECKE / ÖSSZEFOGLALÁS

9.) Logikai feladat:

Fruzsi 5 perc alatt ér az iskolába. Zsuzsinak 10 percig tart az út. Ha ketten együtt mennek, mennyi idő alatt érnek az iskolába?

_____

11.) Gyakorlat:

Az eddig tanultak összefoglalására egy összetett feladatot kell elkészíteni! Tulajdonképpen az első játékot fogjuk programozni!

A programban egy lufival kell eltalálni egy repülő vízilovat! A víziló csak jobbra és ballra repül azonos sebességgel. A lufit egy kis figura fogja elengedni! A figurával jobbra és ballra is lehet lépkedni, és a megfelelő időben elengedni, szóköz megnyomására! A lufi is azonos sebességgel repül felfelé! Ha eltaláltuk a lufival a vízilovat, akkor írjon ki egy szöveget 2 mp-ig! Ha nem találtuk el a lufival, akkor repüljön tovább, ki a képernyőtől, és egy rövid idő eltelte után újra indíthassunk egy újabb léggömböt!

Nézzük az utasításokat:

- A program neve legyen: 16_catch_baloons! A háttér legyen: blue sky!
- Az első szereplő legyen: Gobo!
- A program a zászlóra kattintással induljon!
- A szereplő ugorjon a (0;-140) koordinátára, és a méretét csökkentsük 50% -ra!
- A kiinduló jelmez legyen az „a” állás, és kezdőállapotban mindig nézzen jobbra!
- A kurzormozgató jobb gomb lenyomására menjen 10 képpontot jobbra, ha elérte a szélét, akkor pattanjon vissza, de ne induljon sehova, maradjon ott!
- A bal gomb lenyomására ugyanazokat a műveleteket végezze el, csak az ellenkező irányba! Duplikálj, és változtass az előző programsoron!
- A következő szereplő legyen a Hippo1!
- Készítsd el, hogy zászló lenyomására induljon, a mérete legyen 50 %!
- Majd 0,1 másodpercenként váltson jelmezt, és lépje 10 képpontot!
- Ha a szélére ért pattanjon vissza, és forduljon meg! Folyamatosan repüljön oda-vissza!
- A későbbiekben ennek a szeplőnek az utasításait még bővíteni fogjuk!
- Legyen a harmadik szereplő: Baloon1!
- Zászlóra kattintásra induljon a lufi, és azonnal tűnjön is el, mert majd a „Gobo”-hoz rendeljük!
- Ennek a szereplőnek (lufinak) csak a „c” állását fogjuk használni!
- Ha megnyomjuk a szóközt, akkor jelenjen meg a lufi a „Gobo” szereplőnél, a mérete legyen 50%!
- Ha rákattintottunk a szóköz billentyűre azonnal induljon el a lufi felfelé 0,25 mp-enként, 10 kp-onként!
- Ezt az utasítást 35* hajtsa végre!
- Ha a lufi érintkezik a vízilóval, akkor 0,5 másodperc múlva tűnjön el a lufi!
- Ugyanekkor a víziló mondja: „Nyertél!”, 2 másodpercig!
- A programot úgy készítsük el, hogy ha a lufi nem találja el a vízilovat, akkor repüljön ki a képről fent!
- Egyébként egy kis idő elteltével újabb lufit indíthassunk!
- A program írása közben, és a végén teszteld, javítsd az esetleges hibákat!

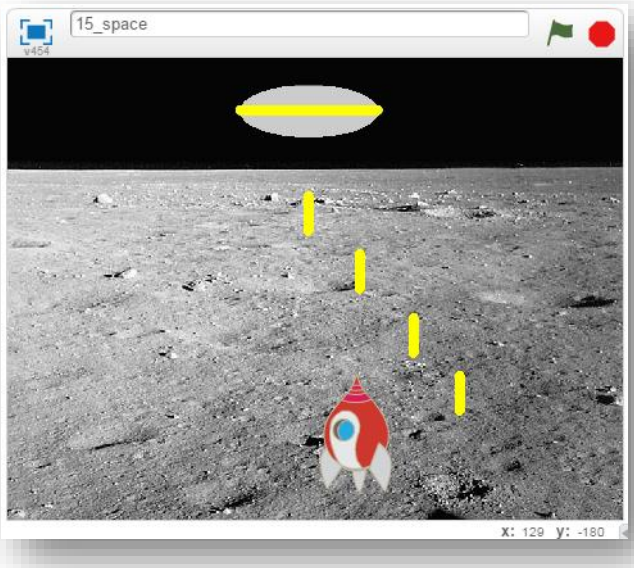
(A megoldást a következő oldalon láthatod! Ne fordíts, amíg nem próbáltad megoldani!)

Baloon1

Godo

Hippo1

8.) Önálló feladat:



Ebben a feladatban egy ufóról tüzelnek egy űrhajóra! Az előző feladattal ellentétben itt nem elkapni kell „szereplőket”, hanem menekülni előle!

A projekt neve legyen: 17_space!

A háttér legyen: moon!

Az ufót neked kell megrajzolni! Két állása legyen! Az alapja egy szürke ellipszis, közepén egy színes vonallal! Az egyik állásban világoskék, a másikban sárga legyen a vonal!

Az ufó jobbra-ballra mozog, és folyamatosan lövi ki a lézersugarakat, melyek sárga rövid vonalak! Ha a sárga sugár eléri az űrhajót, akkor írja ki, hogy veszettél! A szöveg legyen látható 5 másodpercig! Ha a sárga sugár eléri az űrhajót, akkor az ufó ne lőjön ki több sugarat!

A másik szereplő a Spaceship legyen! Melynek az a szerepe, hogy kitérjen a „lézersugarak” elől! Nem csak jobbra és ballra, hanem fel és le is menet, de csak a horizontig! Tehát a fekete színt nem érintheti meg, mert akkor a program dobja vissza a kiinduló helyre az űrhajót! A szereplő mérete az eredeti 50% -a!

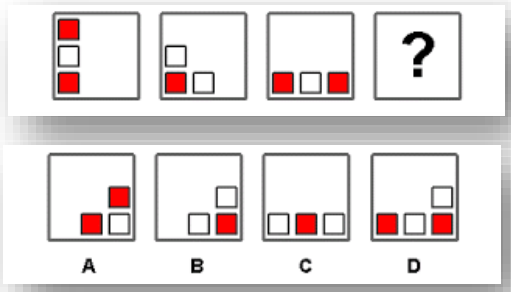
A program írása alatt folyamatosan teszteld a munkádat, és javítsd a hibáidat!

A végén mentsd a megadott helyre a programot 15_space_XX néven! AZ XX a monogramod legyen!

9. LECKE / ÜZENTETEK A SZEREPLŐK KÖZÖTT

10.) Logikai feladat:

Az alsó sorból milyen alakzat kerülhet a kérdőjel helyére



Megoldás: _____

Miért? _____

Az eddigi projektekben mindig a zászlóra kattintással, vagy valamelyik billentyűzet lenyomásával vezéreltük a szereplőket. Ebben a leckében akkor fognak valamit csinálni a szereplők, ha valamilyen üzenetet kapnak, vagy üzenetben kapnak feladatot. A szereplők kommunikálásához szükségünk van az üzenetekre.

Tehát ha az egyik szereplő programjába beillesztem a „küldj üzenetet: -üzenet1- (és várj)”parancsot, akkor a másik szereplő programjának indítása, akkor fog megtörténni, ha az előző programban lefutott ez a rész és megkapja az utasítást!

12.)Gyakorlat:

Ebben a feladatban egy beszélgetést fogunk elkészíteni, úgy hogy a középső szereplő reagál a két másik szereplő utasítására!

Ha a lányra kattintunk bal egérrel, akkor a lány hívja el a fiút discoba, és a fiú forduljon felé, és mondjon: nemet! Ha a jobb oldali fiúra kattintunk, akkor a középső fiú forduljon oda felé, és mondja, hogy: „Hát, persze hogy igen! :)”

Kövessük a következő utasításokat!

- A program neve legyen: 18_talking!
- A háttér legyen: bedroom1!
- Olvass be három szereplőt: Dee-t; Devin-t; és a Maya-t!
- A középső fiúnak az induló jelmeze legyen: „b” állás! Zászlóra kattintással induljon a program!
- A „Dee” szereplőt tedd az ágy végéhez a minta szerint! „Devin”-t tedd a kép bal oldalára! A lányt pedig „ültesd” le az ágy szélére!
- Ha a lány szereplőre kattintunk, akkor mondja 2 másodpercig, hogy „Menjünk discoba!”, és küldjön egy üzenetet a középen álló fiúnak! Az üzenete neve legyen: „lany_mondja”!
- Amikor a középen álló fiú megkapja az üzenetet, akkor a jelmeze legyen „d”, forduljon a lány felé jobbra, várjon 0,5 mp-et! Majd váltson „c” jelmezre, megint várjon 0,5 mp-et! Végül mondja: „Nem! :)”, 2 másodpercig!
- Ha a jobb oldali fiúra kattintunk, akkor jelmeze legyen „b”, de 0,5 mp múlva váltson „a”-ra, és mondja, hogy: „Menjünk el egy focimeccsre!” 2 mp-ig! Küldjön üzenetet: „fiu_mondja” néven!
- Ha megérkezik a középső fiúnak a „fiu_mondja” üzenet, akkor forduljon felé, a jelmeze legyen „e”; várjon 1 mp-et, váltson „a” jelmezre, és végül mondja: „Hát, persze hogy igen! :)”, 1 mp-ig!
- Teszteld a programot közben, és javítsd az esetleges hibákat! Mentsd a megadott néven!
(A megoldást a következő oldalon láthatod! Ne fordíts, amíg nem próbáltad megoldani!)

9.) Önálló feladat:

Ebben a feladatban is az üzeneteknek lesz a fő szerepe!

Az iskola homlokzatán elhelyezett csengőre kattintva indul a program! Először a lány jelenik meg az iskola ajtóban kicsengetés után kisebb méretben, aztán a jobb oldal felé egyre növekedve, mintha kifelé jönne az iskolából.

Amikor kiért forduljon meg, és hívja ki a társát! Ez után jelenik meg az ajtóban a fiú szintén kis méretben, és előre jön a társához, növekedve!

Amikor megérkezik egymás mellé a két szereplő, akkor egy párbeszédbe kezdenek! A párbeszéd végén forduljon meg mind a két szereplő, és hagyja el a helyszínt, ellenkező irányban!

Ebben a projektben, már több szabadságot kapsz, nem lesz annyira kötött, mint eddig! De a minta alapján dolgozz!

Készítsd el a programot következők szerint:

- A program neve legyen: 19_school_finish legyen!
- A háttér legyen: school1 kép!
- A három szereplő legyen: Bell, Avery Walking, Jamie Walking!
- A program lefutása az üzenetekkel legyen szabályozva! A szereplők egymás után adják át a következőnek a parancsokat!
- A párbeszédben minimum 4, maximum 6 mondat váltás legyen!
- A programhoz töltsél le az internetről iskolacsengő hangot, vagy szerezz valahonnan (saját felvétel)! Majd töltsd fel a programhoz, és vágd meg 2-3 másodpercesre! Amikor a csengőre kattintunk az indulásnál, akkor szólaljon meg a csengő is!
- Minden kérdéses esetben szabadkezet kapsz! A sebesség ne legyen túl gyors, se túl lassú!
- Teszteld a programot közben, és javítsd az esetleges hibákat! Mentsd a megadott néven a részfeladatok között is!

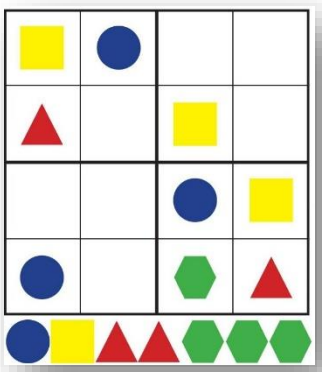
10. LECKE / HÁTTEREK VÁLTOZTATÁSA, ÉS A HÁTTEREKRE SZÖVEG SZERKESZTÉSE

11.) Logikai feladat:

Ebben az egyszerű képes geometriai SODOKU-ban arra kell figyelni, hogy minden sorban, minden oszlopban, és minden vastagon keretezett négyzetben, csak egyszer szerepelhet minden alakzatból egy!

A táblázatot próbáld meg fél perc alatt kitölteni! Mérd stopperrel az időt!

Mennyi idő alatt végeztél? _____ mp



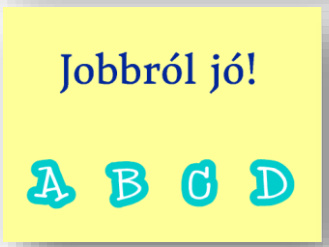
13.)Gyakorlat:

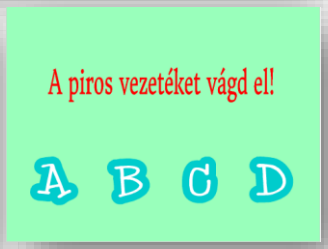
Ebben a feladatban nagyon egyszerű feladatod lesz! Tulajdonképpen programozni nem is kell!

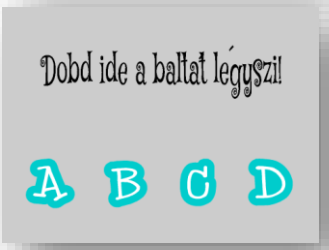
Öt különböző hátteret kell létrehoznod, bizonyos feliratokkal! Ezek között kell lépkedned 4 szereplővel (melyek ebben az esetben az angol abc első négy betűje!

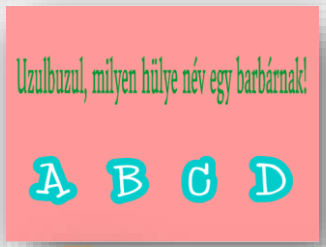
A képernyőminták és az utasítások alapján dolgozz!

- A program neve legyen: 20_last_sentence!
- Szúrjál be összesen 5 db új hátteret! A hátterek neve legyen sorra: ligtgreen; yellow; green; gray; red!
- Mindegyik hátteret színezd a nevének megfelelően valamelyik világosabb színárnyalatra! (vödörrel)
- A képeken pixelgrafikus módban dolgozz!
- A hátterekre különböző betűtípussal (melyekben vannak ékezetes betűk) írd fel a képen látható mondatokat!
- A mondatok minden háttéren legyen más színű!
- Szúrjál be 4 új szereplőt! A minta szerinti betűket!
- A program zászlóra kattintással induljon!
- Old meg, hogy ha egy betűre rákattintunk egérrel, akkor megváltozzon a háttér 5 másodpercre, majd váltson vissza a kiinduló képernyőre!
- Teszteld a programot közben, és javítsd az esetleges hibákat! Mentsd a megadott néven a részfeladatok között is!





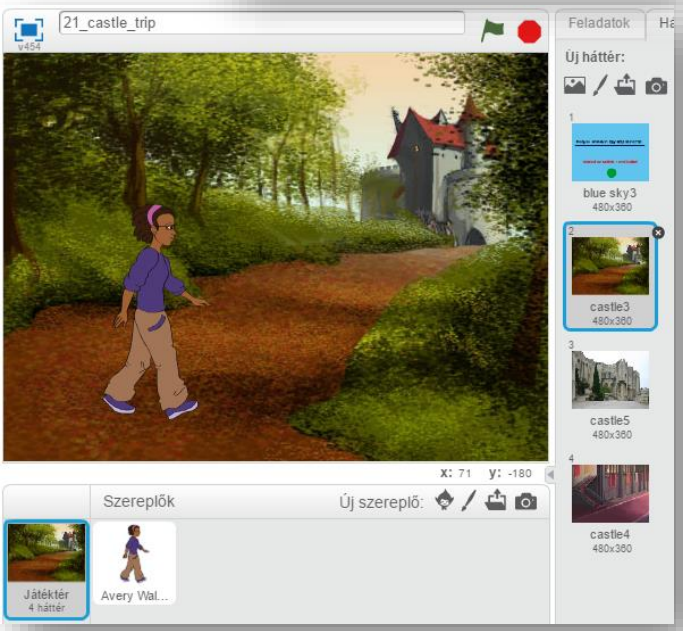




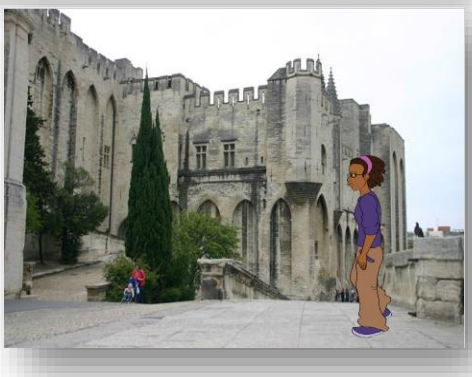
10.)Önálló feladat:

Ebben a programozási feladatban tulajdonképpen egy kisfilmet kell készíteni! Négy darab háttérképet kell váltogatni. A képernyő minták alapján kell elkészíteni a programot!

A program a lányra kattintással indul. A „kisfilmben” a lány sétál a kastély felé az úton, majd a kastély udvarán végig a bejárathoz, aztán az épületen belül felmegy a lépcsőn!



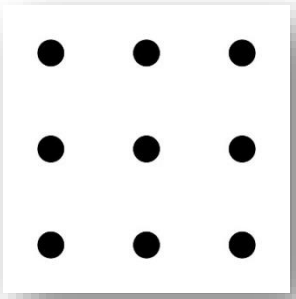
- A program neve: 21_castle_trip legyen!
- Hozzál létre 4 hátteret! Az első legyen egy egyszerű kék lap! A többi pedig sorban: castle3; castle5; castle4!
- Az első képet készítsd el a minta szerint; két világosabb kék téglalappal, és egy sötétkék vonallal! A szöveget, „Gloria”, betűtípussal készítsd, piros színnel!
- A szereplő legyen Avery Walking!
- A második dián az eddig megtanult módon a lány sétáljon a kastély felé egyre kisebb méretben, mintha távolodna!
- Majd váltson a háttérkép, és szintén a lány sétáljon a kastély felé, megfelelő sebességgel!
- Amikor odaért a másik oldalra, újból váltson az utolsó képre! A szereplő jöjjön végig a szőnyegen, majd menjen fel a lépcsőn!
- A legvégén újból ugorjon a kezdő képre!
- Minden kérdéses esetben szabadkezet kapsz, bármely módon megoldhatod!
- Teszteld a programot közben, és javítsd az esetleges hibákat! Mentsd a megadott néven a részfeladatok között is!



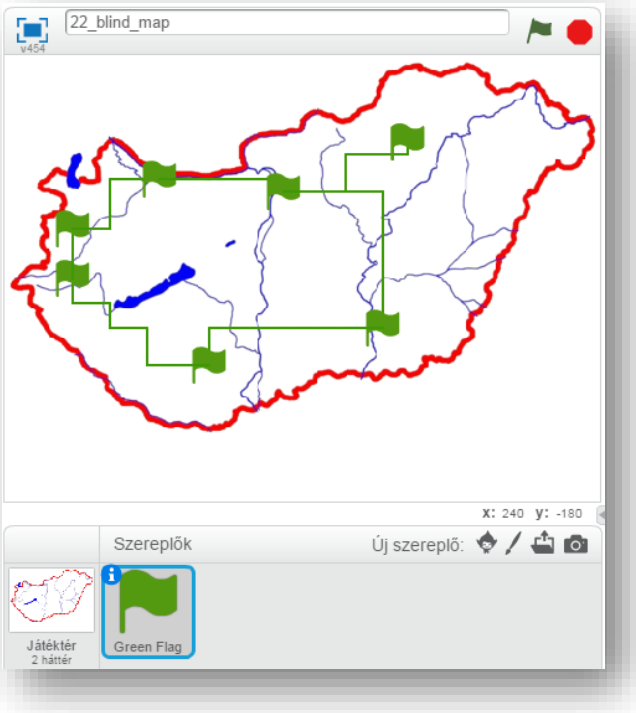
11. LECKE / A „TOLL” PARANCSFAJTÁK CSOPORTJA

12.) Logikai feladat:

Ez a 9 pontos fejtörő. A logikai játék lényege, hogy úgy kell 4 egyenes vonallal összekötni a 9 pontot, hogy a tollat ne emeljük el a papírtól.



A „Toll” parancsokkal, tudunk szereplőkkel lenyomatokat készíteni, tollat letenni, felvenni. Meg tudjuk adni, hogy milyen vastagsággal, milyen árnyalattal, és milyen színnel rajzoljunk. Nézd meg a parancsokat, könnyen rájöhetsz, hogy melyikkel mit tudsz beállítani!



14.)Gyakorlat:

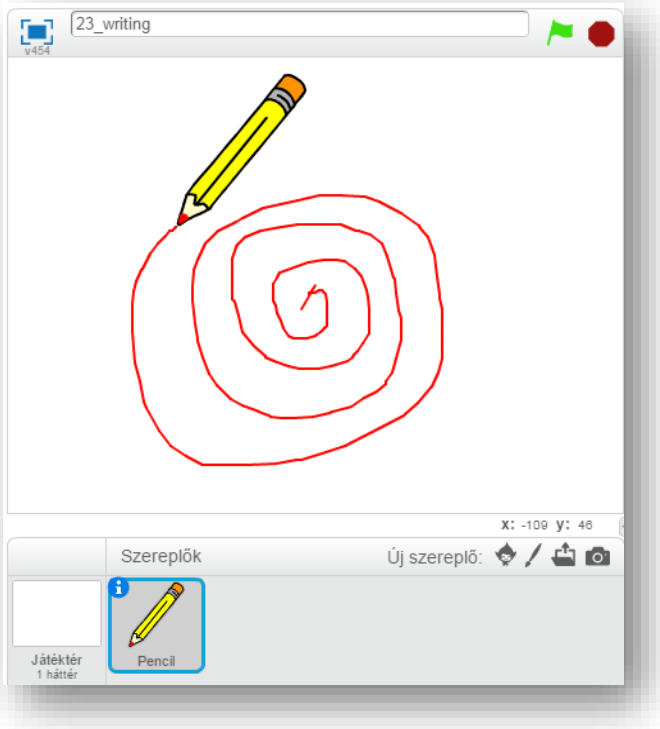
Ebben a feladatban tulajdonképpen a földrajzórán megtanult „vaktérképes” feladatot fogjuk elektronikusan elkészíteni.

(A tényleges földrajzos feladat így hangzik: Juttassunk el a következő városokba rövid útvonalon zöld zászlókat: Eger; Szeged; Pécs; Zalaegerszeg; Szombathely; Győr!)

- A program neve legyen: 22_bind_map!
- A háttér legyen: kep_magyarorszag_terkep! (betölthető, nem eredeti háttér)
- A program törölje az esetleges előzőleg rajzolt dolgokat! Budapestre „leszúrt” zöld zászlóval induljon, melynek a pozíciója (-15;70) (Budapest a térképen)! A zászló mérete legyen lekicsinyítve 60 %-ra!
- Közlekedi a térképen megszokott módon a kurzormozgató billentyűkkel lehessen! Minden irányban a megfelelő „x” illetve „y” koordinátákat 10 képponttal kell növelni illetve csökkenteni!
- A tollat le kell tenni!
- A toll színe legyen zöld színű!
- A toll vastagsága legyen 2 képpont!
- Ha elkészítettük mondjuk a felfelé nyílhoz tartozó programrészt, akkor lehet duplikálni, és csak a gombot kell megváltoztatni, és az irányt! (fel, le, jobbra, ballra) (x,y)
- Ha odaérünk egy városhoz, akkor szóköz lenyomásával tegyünk le egy zászlót! Készíts lenyomatot!
- Teszteld a programot közben, és javítsd az esetleges hibákat! Mentsd a megadott néven! A részfeladatok megoldása között is ments!
  (A megoldást a következő oldalon láthatod! Ne fordíts, amíg nem próbáltad megoldani!)

Egy kicsit nehezebb ellenőrző feladat! Juttassunk el a következő városokba rövid útvonalon zöld zászlókat: Gyöngyös; Debrecen; Székesfehérvár; Veszprém; Békéscsaba; Kecskemét; Miskolc; Kaposvár!

11.)Önálló feladat:



A következő feladatban olyan programot kell készítened, melyben egy fehér lapra piros ceruzával tudsz írni, és rajzolni!

A programot a következő utasítások alapján készítsd el:

- A program neve legyen: 23_witing!
- A szereplő legyen: Pencil!
- Mivel piros színnel szeretnénk rajzolni, a „Pencil” hegyét színezd pirosra!
- A program zászlóra kattintással induljon!
- Minden indulásnál legyen letörölve a rajzfelület!
- Rajzolás, írás a (0;0) koordinátánál kezdődjön!
- Két másodperces várakozással induljon a program! (Azért, hogy odaérjek a zászlótól a ceruzáig!)
- A ceruza kezdőpozícióját úgy állítsad be, hogy a hegyénél írjon majd! (Ezt a jelmez kezdőkeresztre való arrébbhúzásával lehet megoldani!)
- A rajzolószín legyen piros (1)!
- A vonal vastagsága legyen 2!
- A program úgy fog működni, hogy az egeret követve folyamatosan rajzol!
- Teszteld a programot közben, és javítsd az esetleges hibákat! Mentsd a megadott néven!

12. LECKE / VÉLETLEN SZÁMOK GENERÁLÁSA, ALKALMAZÁSA PROGRAMOKBAN

13.) Logikai feladat:

Rajzold a jobb alsó téglalapba az odaillő ábrát!

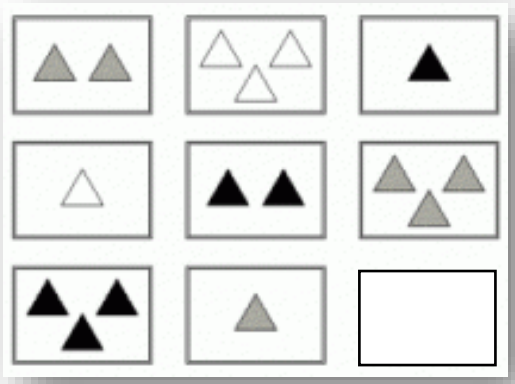
Miért? Magyarázd meg röviden!

_____

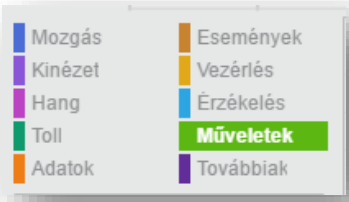
_____

_____

_____



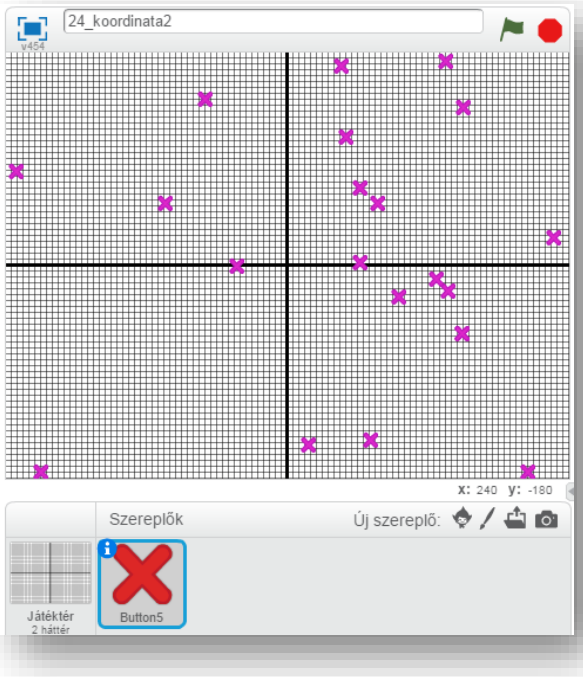
Már majdnem mindegyik „parancsfajták csoport”-ját megnéztük, és válogattunk a benne lévő lehetőségek közül. Most a „Műveletek” csoportból a véletlen szám generátort fogjuk kipróbálni. Ez az egyik legfontosabb, legtöbbször használt parancs a programozásban!



A Scratch-ben 

parancsot használjuk!

15.)Gyakorlat:

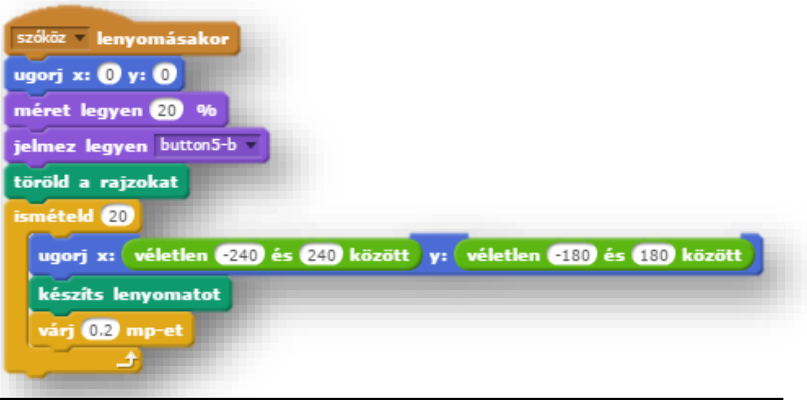


Először tulajdonképpen egy nagyon egyszerű programban használjuk a véletlen parancsot!

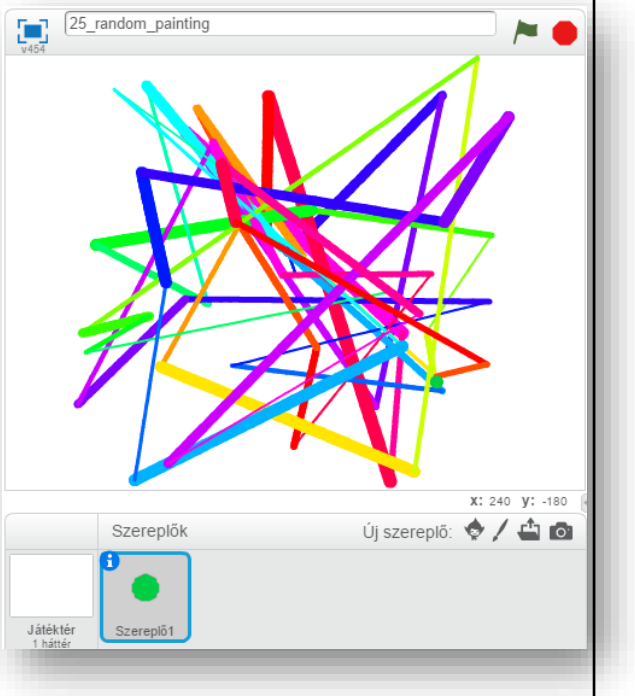
Egy beolvasott koordináta tengelyen véletlenszerűen helyezzen el 20 darab „X”-et!

- A program neve legyen 24_koorinata2!
- A háttér legyen: kep_koordinata_tengely!
- A program szóköz lenyomására induljon!
- A szereplő legyen a Button5, „b” állása, és az eredeti méret 20%-a!
- Minden program futtatásakor törölje az előzőleg letett szereplőket!
- A program lefutásakor hússzor készítsen lenyomatot véletlenszerű helyeken!
- Ezt az „ugorj x: ___ y: ___” paranccsal érhetett el! Ennek a parancsnak a fehér helyére húzhatod be a véletlen parancsot!
- Mivel a rajzfelület 480-360-as ezért az x-et -240-től, +240-ig teheted le. Az y-t pedig -180-tól; +180-ig!
- A véletlen koordináta generálása után lenyomatot készítünk az adott pozícióba!
- Majd, hogy ne fusson le nagyon gyorsan, késleltessük a szereplő letételét 0,2 mp-el!
- Teszteld a programot közben, és javítsd az esetleges hibákat! Mentsd a megadott néven!
  (A megoldást a következő oldalon láthatod! Ne fordíts, amíg nem próbáltad megoldani!)

Az előző oldalon lévő feladat megoldása:



12.)Önálló feladat:



Ebben a rövid programban véletlen színű, véletlen vastagságú folytonos vonalat húz egy fehér felületre, a koordináta véletlen pontjára csúszva egy másodpercenkénti váltással.

Takard le a megoldást, hogy csak az utasítások alapján haladj!

- A program neve legyen: 25_random_painting!
- A program szóköz lenyomására induljon!
- Hozzunk létre egy új szereplőt, amely legyen egy kis zöld kör!
- Az elején mindig töröljük az előző futtatás eredményét!
- A rajzolás (0;0) koordinátáról induljon!
- A programrész folyamatosan ismétlődjön, amíg le nem állítjuk!
- A toll mérete kiinduláskor legyen: 1!
- Tegyük le a tollat a rajzoláshoz!
- A rajzolószín véletlenszerűen változzon 1 és 100 közötti értéken!
- A toll mérete is véletlenszerűen változzon 1 és 10 között!
- Végül érjük el, hogy 1 másodperc alatt érjen a vonal véletlenszerűen a rajzfelületen valahova!
- Teszteld a programot közben, és javítsd az esetleges hibákat! Mentsd a megadott néven!

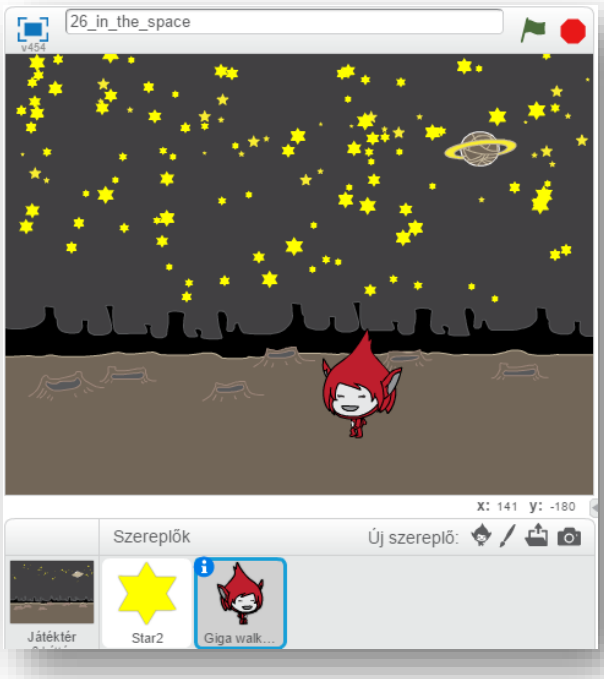
13. LECKE / ÖSSZEFOGLALÁS

14.) Logikai feladat:

Milyen szám illik a kérdőjel helyére?

5 (3) 8 (1) 9 (4) 13 (5) ? Megoldás: _____

13.)Önálló feladat:



Foglaljuk össze az előző leckékben tanultakat egy programban!

A feladatot a következő utasítások alapján készítsd el:

- A program neve: 26_in_the_space legyen!
- A program szóköz lenyomására induljon!
- Két részből fog állni a feladat!
- A szereplők „Star2” és „Giga walk” legyen!
- Ezek a szereplők legyenek rejtve, szóköz lenyomásakor!
- A programban két háttérképet fogunk használni! A „Stars” és a „Space” képeket!
- A nyitóképernyő legyen a „Stars” kép!
- Rajzolj egy kisebb szürke kör szereplőt, és nevezd el „planet”-nek!
- A szürke kört helyezzük a (0,0) koordinátára!
- Ez a kör növekedjen, hogy úgy nézzen ki, mintha közelednénk felé! 10 képpontonként növekedjen! 0,2 mp-enként, és fusson le 20-szor!
- Amikor megnőtt, írasd ki a „Nézzünk körül!” szöveget 2 másodpercig!
- Ezek után váltsunk a másik képre, és kezdjük a másik részét a programnak!
- Üzenet küldésével oldjuk meg a váltást!
- Rejtsük el a szürke kört, és jelenítsük meg a többi szereplőt!
- Két különálló szereplő véletlenszerű feladatait kell elkészíteni!
- Az „égen” 100 darab csillagot kell véletlenszerűen elhelyezni különböző méretben.
- A „földön” pedig véletlenszerű pozícióba csússzon a szereplő!
- Minden indításkor törölje az előző futtatás rajzait!
- 100 darab csillagot helyezz véletlenszerűen elhelyezni az égboltra, a méret legyen 5% és 30% között!
- A lenyomatkészítés után várakozzunk 0,1 másodpercet, hogy ne legyen túl gyors!
- A másik szereplő 60%-os méretben sétáljon a földön!
- Véletlen helyre csússzon, és váltson jelmezt!
- Ha a szélére ér pattanjon vissza, és váltson nézetet balról jobbra!
- Majd itt is várj 0,1 másodpercet!
- Ha lefutott ez a programrész, akkor minden álljon le!
- Teszteld a programot közben, és javítsd az esetleges hibákat! Mentsd a megadott néven!

14. LECKE / MATEMATIKAI, RELÁCIÓS ÉS LOGIKAI MŰVELETEK HASZNÁLATA A SCRATCH-BEN

15.) Logikai feladat:

Számold ki, hogy mi kerül a kérdőjel helyére!

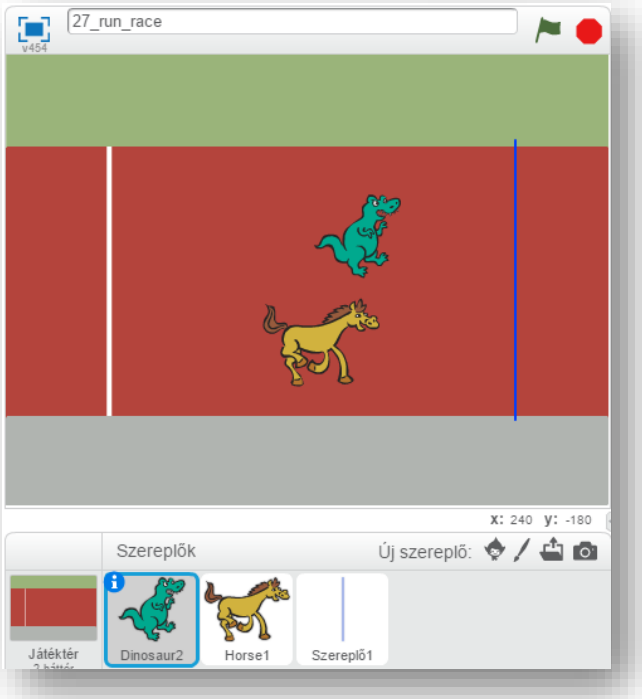
Megoldás ? = _____

Körte: _____ Cseresznye: _____

A következő műveleteket a programozásban nagyon sokszor fogjuk használni:

- Összead két számot.
- Kivonja az első számból a másodikat.
- Összeszoroz két számot.
- Elosztja az első számot a másodikkal.
- Igaz értéket ad, ha az első szám kisebb, mint a második (vagy az első szöveg betűrendben előrébb van, mint a második).
- Igaz értéket ad, ha a két szám egyenlő (vagy a két szöveg megegyezik).
- Igaz értéket ad, ha az első szám nagyobb, mint a második (vagy az első szöveg betűrendben hátrébb van, mint a második).
- Az ÉS igaz értéket ad, ha mindkét feltétel igaz.
- A VAGY igaz értéket ad, ha valamelyik feltétel igaz.
- A NEM igaz értéket ad, ha a feltétel hamis, és hamisat, ha igaz.

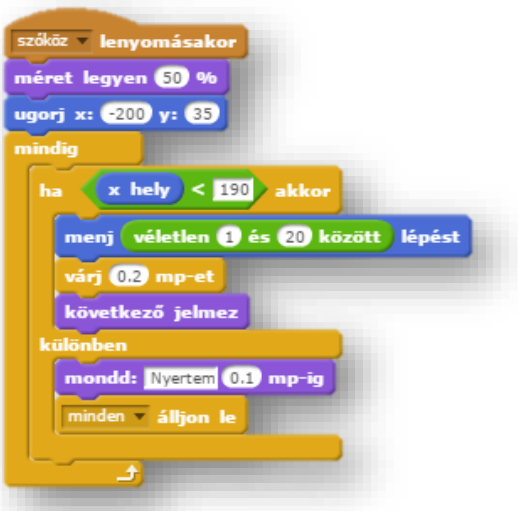
16.)Gyakorlat:



Ebben a programban egy ló és egy dinoszaurusz versenyt fut, véletlen generált lépésnagysággal, így nem tudni ki lesz a nyertes! Amelyik beért a célba, az elkiáltja magát, hogy „Nyertem! Majd leáll a program!

- A program neve legyen: 27_run_race!
- A háttér legyen: track!
- A két szereplő: Dinosaur2; Horse1 legyen; és készíts egy kék vonalat a pálya végére!
- Mind a két szereplőnek körülbelül ugyanazt kell tennie!
- A méretük legyen 50%!
- A kezdő pozíció a fehér vonal mögött legyen!
- A programot úgy készítsd el, hogy a szereplő menjen véletlenszerűen 1és 20 kp közötti lépsekkel előre, és ha elérte az x koordináta a 190 kp-ot, akkor mondja, hogy „Nyertem!” 0.1 mp-ig, majd minden álljon le!
- Teszteld a programot közben, és javítsd az esetleges hibákat! Mentsd a megadott néven!

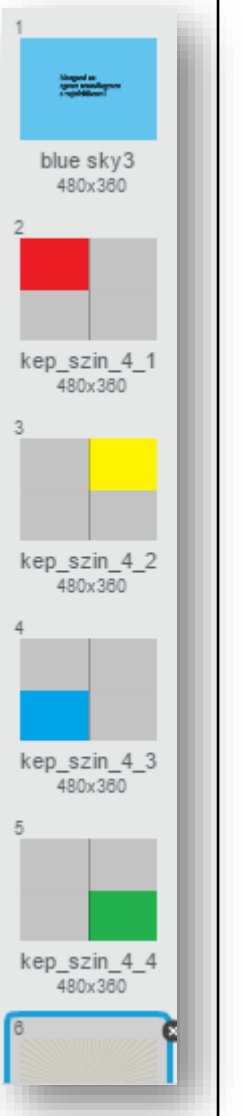
Az előző oldalon lévő feladat magoldása. A képpontok különbözősége a szereplők méretének különbségéből adódik!



14.) Önálló feladat:

Ebben a programban az egér mozgását fogjuk figyelni, és attól függően, hogy hol tartózkodik, változtatjuk a hátteret!

- A program neve legyen: 28_mouse_move!
- Két beépített hátteret olvass be! AZ egyik legyen a „blue sky”, a másik legyen a „ray”!
- Az elsőre írd rá, hogy „Mozgasd az egeret tetszőlegesen a rajzfelületen! Tetszőleges betűtípust használhatsz!
- A másikra középre írd rá, hogy „Vége!”
- Olvass be tallózással: kep_szin_4_1; kep_szin_4_2; kep_szin_4_3; kep_szin_4_4 képeket!
- A program szóköz lenyomására induljon!
- A háttér legyen „blue sky”!
- Várjon 5 másodpercet, és nullázzad le az időmérőt!
- Vizsgáld az egér pozícióját, hogy melyik negyedben mozgatjuk! Attól függően változzon a megfelelően háttér!
- 10 másodperc múlva automatikusan váltson a „ray” háttérre és álljon le minden!
- Teszteld a programot közben, és javítsd az esetleges hibákat! Mentsd a megadott néven!



15. LECKE / A VÁLTOZÓK LÉTREHOZÁSA, HASZNÁLATA A SRATCH-BEN

16.) Logikai feladat:

Milyen számok kerüljenek az üres négyzetekbe, hogy a szorzatuk vízszintesen, függőlegesen és átlósan is mindig nyolc legyen?

Írd az üres négyzetekbe a megfelelő számokat!

2		
	2	**1**
		2

Az „Adatok” csoportnál „Változókat” és „Listákat” hozhatunk létre. Ebben a leckékben csak a változókkal foglalkozunk.

- A változó neve melletti szürke négyzetre kattintással a változó kijelzőjét eltüntetheted, vagy megjelenítheted a játéktérben.
- A megjelenő kijelzőt bárhová mozgathatod, és a jobb kattintással előhívható helyi menüben a normál és a nagy megjelenítés között is választhatsz.
- Választhatod továbbá a csúszka megjelenítése opciót, amivel lehetőséget adhatsz a programot használónak arra, hogy a változó értékét saját maga változtathassa meg a program futása alatt. Egy újabb jobb kattintással beállíthatod a csúszka lehetséges legkisebb és legnagyobb értékét is a csúszka határainak megadásával.
- Beállítja a változó értékét a megadott értékre.
- Megváltoztatja a változó értékét a megadott értékkel (ha az nagyobb nullánál, akkor növel, ha kisebb mint nulla, akkor csökkent).
- Megjeleníti a változó értékét a Játéktérben.
- Eltünteti a változó értékét a Játéktérből.

17.)Gyakorlat:

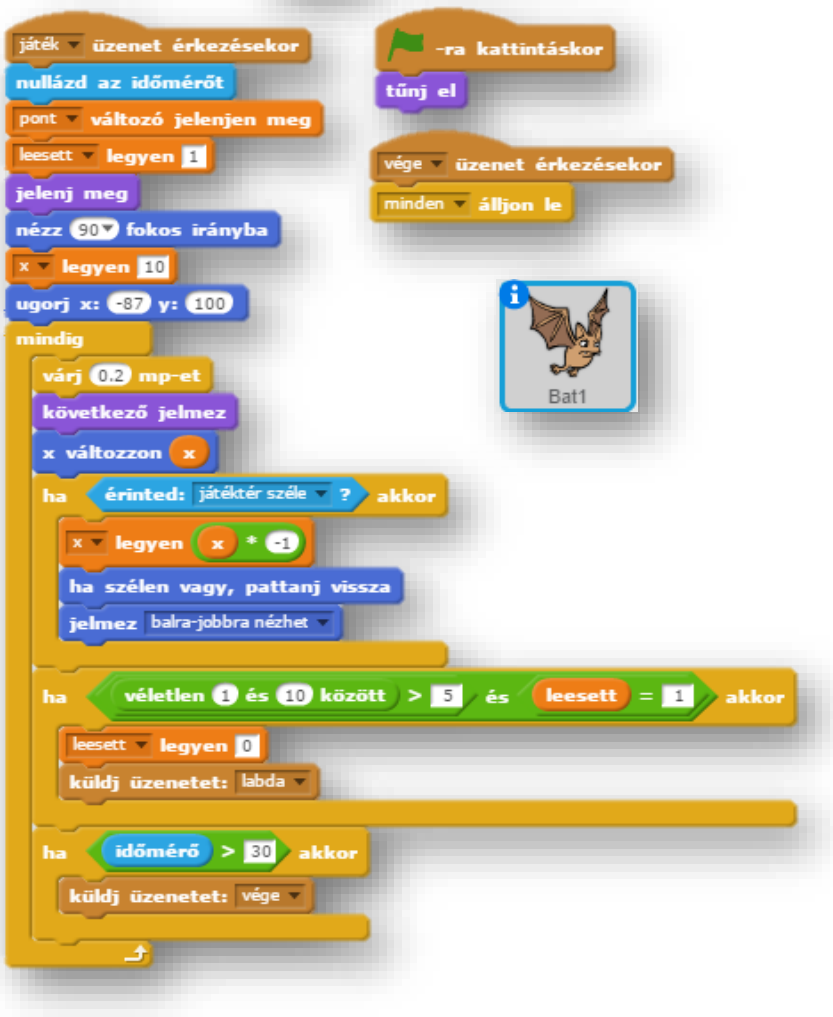
Elérkeztünk arra a szintre, hogy az első komoly játékprogramunkat el tudjuk készíteni!

Képernyőminta:

A játékban a denevér repked a kép felső részén, és almákat dobál. Alul egy tállal el kell kapni az almákat. Pontokat gyűjtünk az elkapott almákkal. Beállítunk egy időkorlátot, amely 30 másodperc!

- A program neve legyen: 29_bat_game!
- A két háttér legyen „blue sky” és „doily”!
- A „doily”-t duplikáld, és a másodikra írd rá „Gloria” betűtípussal, hogy „Nagyon szép eredmény!!!” szöveget, kék színnel!
- A négy szereplő legyen: Butterfly1; Bat1; Apple; Bowl!
- A program zászlóra kattintással induljon!
- A programban két üzenetet is használni fogunk, ezeket hozzuk létre! Legyen egy „játék” nevű, és legyen egy „vége” nevű!
- A kezdőképernyő legyen a „doily”!
- A pillangó jelenjen meg a képernyő bal oldalán, és repüljön be közére, álljon meg és mondja, hogy „Gyűjts minél több almát 30 másodperc alatt!
- Ezután küldjön üzenetet, hogy kezdődjön a játék, és a pillangó tűnjön el!
- A tálhoz tartozó parancsokat készítjük el a következőkben!
- A program indításakor elrejtjük a szereplőt!
- Viszont amikor érkezik a „játék” üzenet akkor láthatóvá tesszük!
- A tálka mindig felfelé néző állapotban legyen!
- Készítsünk ciklust, hogy ha jobbra gombot nyomjuk meg, akkor 10 képpontot változzon az „x” koordináta, ha pedig a bal gombot nyomjuk meg, akkor változzon -10 képpontot az „x” koordináta!
- Ha a széléhez ér a szereplő, akkor pattanjon vissza!
- Ha „vége” üzenet érkezik, akkor tűnjön el a szereplő!
- A feladat megoldásához változókat kell létre hozni!
- Három változó szükséges a megoldáshoz!
- A „leesett” változó számolja majd az el nem kapott almákat!
- A „pont” változóval számoljuk az elkapott almákat!
- Az „x” változóval koordináta értéket fogunk tárolni, melyet felhasználunk a mozgásoknál!
- Az alma szereplő parancsainál komolyabb programozási gondolatmenetre lesz szükségünk!
- A program indításakor a szereplő ne legyen látható!
- A „pont” változót le kell nullázni, hogy majd egyesével növelni tudjuk a találatokkor!
- A „labda” üzenet érkezésekor induljon a programrészlet!
- A szereplő ugorjon a denevér helyére, mert az a szereplő fogja „eldobni”!

- Aztán jöjjön egy ciklus, melyben az alma „y” koordinátáját -10 képponttal csökkentjük!
- Ha a szereplő érinti a játéktér szélét, alját, leesik a földre, akkor tűnjön el a szereplő, és a „leesett” változót növeljük eggyel!
- Ha a tálat érintjük a szereplővel, akkor „pont” értékét növeljük meg eggyel! Ezt úgy tudjuk megtenni, hogy „pont:=pont+1” parancsot használjuk a mint szerin! (Ezt jegyezzük meg, mert ilyet sokszor fogunk használni, programozáskor!)
- A „leesett” itt is legyen 1, mert erre az értékre szükség lesz!
- „Vége” üzenet érkezésekor legyen láthatatlan a szereplő!
- A denevér szereplő, a program indulásakor, legyen rejtve! „Játék” üzenet érkezésekor induljon!
- Nullázzuk le az időmérőt!
- A „pont” változó jelenjen meg a jobb alsó sarokban, nagy nézetben!
- A „leesett” változó értékét nullázzuk le!
- A denevér mindig nézzen 90 fokban felfelé irányba!
- Az „x” változónak adjunk értéket! Legyen 10!
- Ugorjon a szereplő a kezdő pozícióba (-90,100)!
- Majd jön egy hosszú ciklus, melyben ismétlődnek a következő utasítások!
- Kicsit késleltetjük a szereplő mozgását 0,2 mp-el!
- Jelmezt váltunk!
- Beállítjuk, hogy az „x” koordináta változzon az „x” változó” aktuális értékével, amit az előző lefutásból hozott, vagy első lefutáskor a program elején megkapóból vesz!
- Ha a denevér eléri a játéktér szélét, akkor negatív előjellel visszafelé indítjuk! Ezt előjelváltással érjük el! X:=X*(-1)! (10 ⇨ -10; -10 ⇨ 10)
- A jelmez irány szerint váltjuk!
- Az alma véletlen szerű elengedését a következő módon oldjuk meg: generálunk egy véletlen számot 1 és 10 között, ha ez nagyobb mint 5; és a „leesett” változó értéke 1, akkor lenullázom a leesett változót és küldök egy üzenetet, hogy labda!
- Ha letelt a 30 mp, akkor küldjön vége üzenetet, és álljon le minden!
- Teszteld a programot közben, és javítsd az esetleges hibákat! Mentsd a megadott néven!



15.)Önálló feladat:

A következő játékprogram nagyon hasonlít az előző feladathoz! A képernyő tervek alapján kell elkészítened a feladatot! Nem kapsz semmilyen egyéb instrukciót, csak a következőket!

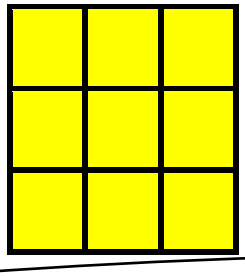
- A program neve legyen: 30_balloon_game!
- A polip engedjen el lufikat, a papagáj kapja el, és ezért kapjunk pontokat! A lufik felfelé szállnak!
- A megfelelő háttereket keresd ki. és használd!
- Teszteld a programot közben, és javítsd az esetleges hibákat! Mentsd a megadott néven!

16. LECKE / MŰVELETEK VÁLTOZÓKKAL / PROGRAMOZÁSI ALAPMŰVELETEK

17.) Logikai feladat:

Hány sárga négyzetet látsz ezen a képen? Számold össze!

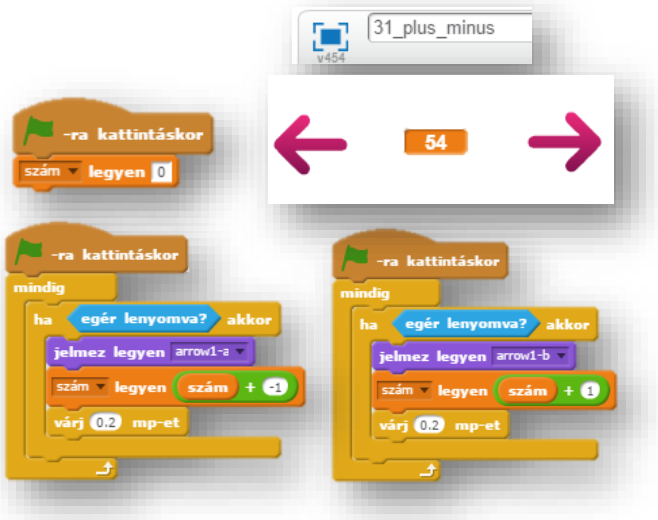
Megoldás: _____



18. Gyakorlat: Ebben a leckében fontos programozási alapdolgokat fogunk megnézni, amelyekkel bármely programnyelv használatakor találkozhatunk. Ezeket a műveletekhez változókat használunk!

a.) Egy **változó értékének növelése, csökkentése** eggyel.

- A program neve legyen: 31_plus_minus
- Hozzunk létre egy változót „szám” néven! Nagy nézetet állítsunk be jobb egérrel!
- Két „Arrow” szereplőt használjunk! Csak két jelmezét használjuk, az „a” illetve a „b” állását! A két nyilat tegyük a változó két oldalára!
- A program a „mindig” vezérlő elemmel folyamatosan vizsgálja, az egér kattintását!
- Ha kattintottunk az egérrel a ➔ szereplőn, akkor a „szám” változó legyen „szám”+1! Ha kattintottunk a ➔ szereplőn akkor „szám” változó legyen „szám”-1.



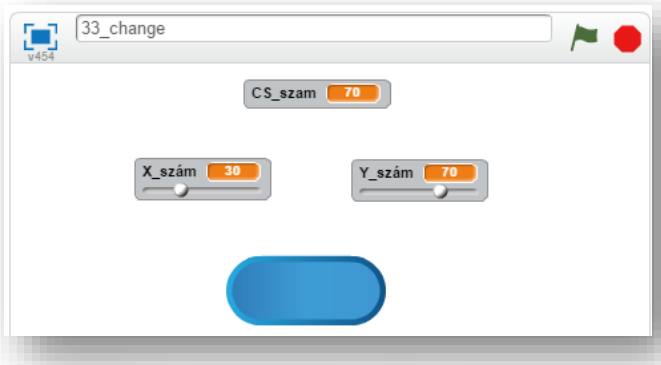
Tehát általánosságban, ha növelni akarom egy változó értékét eggyel, akkor a „v:=v+1” (változó legyen egyenlő változó + 1)!

b.) Az egyszerű **matematikai műveletek** végzésénél is vannak dolgok, amire figyelni kell!

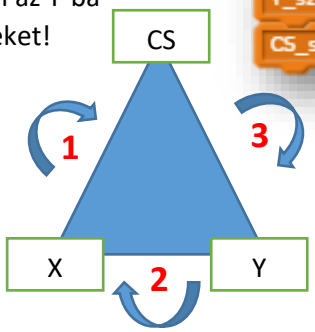
Két változót fogunk „beolvasni”, és ezekkel végezzük el az alapműveleteket!

- Hozzunk létre egy 32_calculate nevű programot!
- Hozzunk létre két változót „a_szam”, és „b_szam” néven!
- Scratch-ben úgy tudjuk a számok beolvasást megoldani, hogy „csúszkává” alakítjuk a változót! Jobb egér lenyomásával lehet kiválasztani! 1-100-ig lehet számokat állítani! Jobb egérrel, megadhatunk más min. és max. értékeket a csúszkának!
- Minden programnyelvnél, a program elején be kell állítani a kezdőértéket minden változóra! (Komolyabb programnyelveknél a típusokat is be kellene állítani, de itt nem kell!) Mindent lenullázunk, kivéve a „szorzat” változót, mert annak „1” –et kell adni. (0 –val való szorzás miatt!)
- Majd egy folyamatosan futó „mindig” ciklusban egyenlővé tesszük a változókat a hozzájuk tartozó műveletekkel a minta szerint!

c.) A változók értékének felcserélése!



- A program neve legyen: 33_change!
- Hozzunk létre három változót! „X_szám; Y_szám; és CS_szám néven! A „CS_szám” változóra azért van szükség, hogy ideiglenesen itt tároljuk az egyik változót!
- Vegyünk fel egy „Button2” szereplőt!
- A feladat maga az, hogy a z X és Y szereplőt felcseréljük!
- Az elején itt is lenullázzuk a változókat!
- Aztán jöjjön a csere!
- Az ideiglenes változóba beletesszük az X-et!
- Az Y-ból az értéket áttesszük az X-be!
- Végül az ideiglenes tárolóból az Y-ba áttesszük az abban lévő értéket!



X → CS

Y → X

CS → Y

---

d.) Maradékos osztásnak hívják a programozásban a „mod” parancsot!

- A **„mod”** parancsnak az a lényege: ha a „Szám” változót elosztjuk az „Osztó” változóval, akkor az eredmény egész részével nem foglalkozunk (arra ott a „div” parancs), hanem maradékot adja vissza egész számként!

Pl.: Eredeti: 8/5=1,6 8 mod 5 = 3
A nyolcat osztjuk öttel, akkor meg van benne egyszer, és a maradék a három! A mod-nál a 3 az eredmény

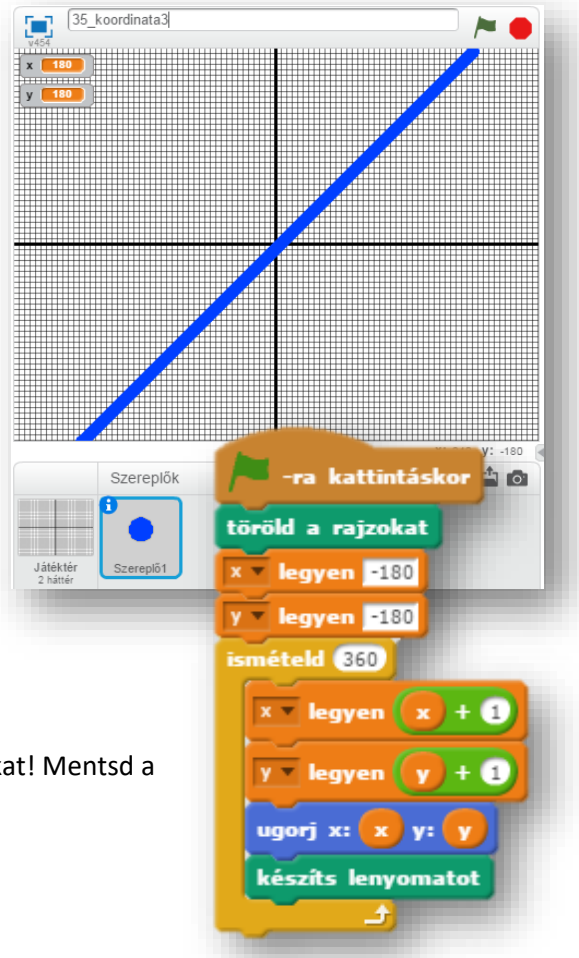
Még egy példa:
Eredeti: 20/6=3,33 20 mod 6 =2
Tehát a húszat elosztjuk hattal, az három, de nekünk a maradék kell az pedig a 2!

- A **„kerekítés”** parancs egészre kerekít egyértelműen!
- A **„gyök”** parancs, és a legördülő menüből is magától értetődő választási lehetőségeink vannak.
- **„Sin; cos; tg; log; abszolút érték; felfelé kerekítés; lefelé kerekítés;”** stb… közül választhatunk!

19. Gyakorlat:

A programban egy egyenes vonalat fogunk kirajzolni a koordináta tengelyen, ahol egy ciklusban változtatjuk az „x” és az „y” értékét! Mindig eggyel növeljük!

- Hozzunk létre egy 35_koordináta3 nevű programot!
- A háttér legyen a kep_koordináta_tengely nevű jpg képfájl!
- Hozz létre szereplőnek egy egyszerű kék kis kört! Ennek a mozgása fogja kirajzolni az egyenes vonalat!
- A programot zászlóra kattintással indítjuk!
- Hozzunk létre két változót, egy „x” –et és egy „y” –t!
- Az alakzatot (-180;-180) koordinátából indítjuk!
- Töröljük az esetleges előzőleg futtatott program maradvány rajzait!
- Az „x” és az „y” értékét egyesével növeljük, és készítsünk lenyomatot a szereplőről!
- 360 szor ismételjük, mert -180 tól +180-ig rajzolunk!
- Teszteld a programot közben, és javítsd az esetleges hibákat! Mentsd a megadott néven!

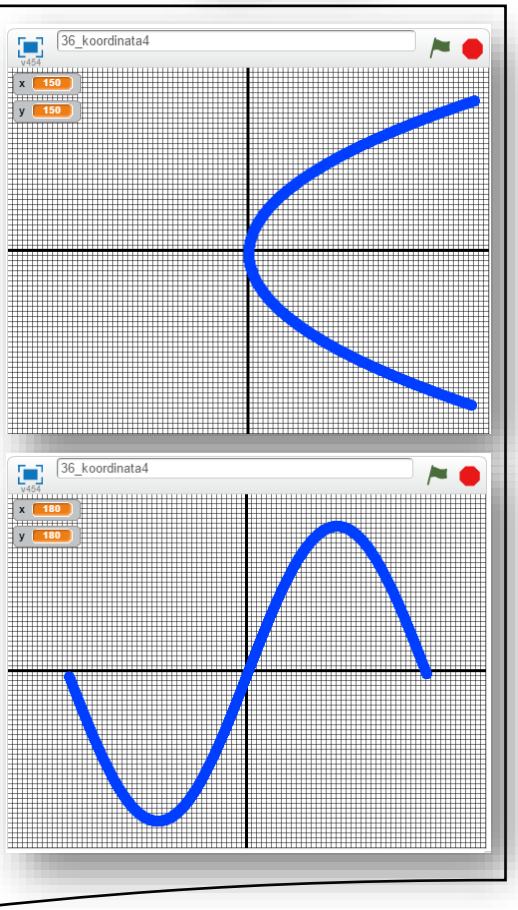


16.)Önálló feladat:

Az előző feladatot csak pár sorban kell megváltoztatni, ahhoz, hogy ezt két vonalat rajzoltasd ki a képernyőre!

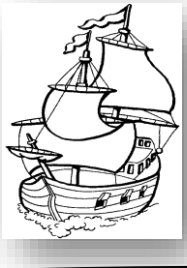
a.) Eredetileg a feladat, az x^2 kirajzolása! De a nagy számok miatt, változtass rajta $x^2/100$ –ra.
- A program neve legyen 36_koordinata4_a!
- Az „x” és az „y” is -150 –ból induljon!
- Ugyanúgy növeld az „x” és az „y” -t is, de a kiíratásnál, amikor az „ugrást” adod meg, akkor illeszd be az $x^2/100$ képletet!
- 300 szor ismételje a műveletet!
- Teszteld a programot közben, és javítsd az esetleges hibákat! Mentsd a megadott néven!

b.) Itt a feladat, egy sinus görbe kirajzolása a képernyőre!
- A program neve legyen 36_koordinata4_b!
- Az alakzatot (-180;-180) koordinátából indítjuk!
- 300 szor ismételje a műveletet!
- Ugrásnál az „x” maradjon változatlan, viszont az „y” legyen „sin(y)*150”! (Itt is változtatunk a méreten.)
- Teszteld a programot közben, és javítsd az esetleges hibákat! Mentsd a megadott néven!



17. LECKE / LISTA LÉTREHOZÁSA ÉS HASZNÁLATA

18.) Logikai feladat:



Minden nap délben egy hajó indul el San Franisco-ból Pearl Harbor-ba a Csendes óceánon, ez a hajózási társaság ugyanebben a pillanatban New Yorkból Le Havre-ba is indít járatot. Az útvonalat mindkét irányban pontosan hét nap alatt teszik meg a hajók. Egy induló hajó hány szembejövővel találkozik? (Ha kell, rajzold le!)

Megoldás: _____

A listákban számokat vagy szövegeket tárolhatunk sorban egymás után. Új elem hozzáadásakor az elem a lista végére kerül, tehát ez a szerkezet hasonlít egy egyszerű bevásárló listához: ha eszünkbe jut valami, amit venni kell, a lista végére írjuk.

Listákat is a változók csoportban hozhatsz létre a „Lista létrehozása” gombbal. A létrehozás ugyanúgy történik, mint változók esetén. A lista létrejötte után megjelennek a listakezelő parancsok:

Gyorsan egy litát úgy tudunk feltölteni, hogy a projekt felületen megjelenő kis ablak bal alsó sarkában lévő „+” jelre kattintunk, és beírjuk a kívánt szöveget, vagy számot!

A lista ablakát, a jobb alul található vonalas résszel tudjuk mértezni.

- Megadja a lista összes elemét.
- Felveszi a lista végére a megadott értéket. Az érték lehet szám vagy szöveg.
- Törli a lista valamelyik, vagy az összes elemét. A legördülő menü utolsó pontját választva a lista utolsó elemét törli. A minden pontot választva a lista összes elemét törli. A törlés csökkenti a lista hosszát.
- Beszúrja a lista megadott helyére a megadott értéket. A legördülő menü utolsó pontját választva a lista végére veszi fel az értéket. Az egyik pontot választva a lista egy véletlenszerűen választott helyére szúrja be az értéket. Elem beszúrásakor a korábbi elemek megmaradnak, és a lista hossza eggyel nő.
- Lecseréli a listában a megadott helyű elemet a megadott értékre. A legördülő menü utolsó pontját választva a lista utolsó elemét cseréli le. Az egyik pontot választva a lista egy véletlenszerűen választott elemét cseréli le. Elem lecserélésekor a lista hossza nem változik meg.
- Megadja a lista megadott sorszámú elemét. A legördülő menü egyik pontját választva a lista egy véletlenszerűen választott elemét adja meg.
- Megadja a lista elemeinek számát.
- Igaz értéket ad, ha a listában a megadott elem megtalálható (csak pontos egyezés esetén ad igaz értéket).
- A projekt készítésekor elrejthetjük, vagy láthatóvá tehetjük a listát!

20.) Gyakorlat:

Ebben az egyszerű programban egy üres listát fogunk feltölteni 1 és 100 közötti véletlen számmal!

- A project neve legyen 37_random_number_list!
- Hozzunk létre egy számok nevű listát!
- A program a játéktérre kattintással induljon!
- Töröljük le az esetleges előző futtatáskor létrehozott elemeket!
- A lépéseket késleltessük egy kicsit, hogy jobban lássuk a műveleteket! Ezért a törlés után várjunk 1 másodpercet!
- 10 számot szeretnénk generálni ezért az „ismételd” parancsot alkalmazzunk!
- Az ciklusban a „számok” listába adjunk hozzá egy 1 és 100 közötti véletlen számot! Ezt az utasítás végrehajtást is késleltessük egy kicsit!
- Mentsünk, és futtassuk!

21.) Gyakorlat:

A következő projektet más módon, egyszerűbben is meg lehetne oldani, de mivel a listákat gyakoroljuk, nézzük meg így!

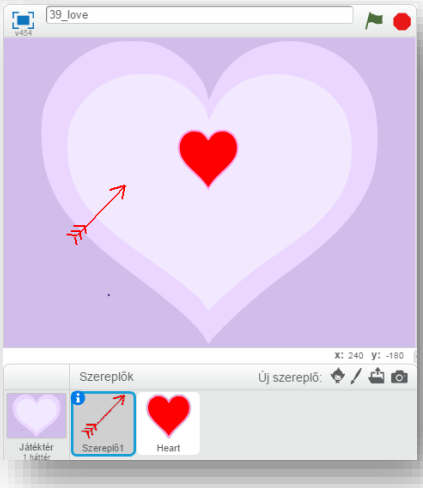
Egyszerűen annyi a feladat, ha az egérrel megérintjük a dobókockát, akkor véletlenszerűen írjon ki a listából egy elemet!

- A program neve legyen:38_drop!
- Olvasd be szereplőnek a kép_dobokocka.jpg-t!
- Zászlóra kattintással induljon a program!
- A fehér háttérre írd fel zöld betűkkel: „Érintsd meg a kockát!”
- Ha megérintjük a kocka szereplőt, akkor véletlenszerűen írjon ki egy számot a „dobás” listából!
- Mentsünk, és futtassuk!

17.)Önálló feladat:

Ebben a feladatban a középen lévő piros szívet egérrel érintve, véletlen pozíciókba nyilakat szúrunk, és 2 másodpercre nevek jelennek meg a nyíl szereplő mellett!

- A program neve legyen: 39_love!
- A háttérkép a hearts1 legyen, és olvasd be a Heart szereplőt!
- Rajzolj egy új szereplőnek egy piros nyilat a minta alapján!
- Készíts egy listát „nevek” néven
- A szív szereplő érintésekor véletlen koordinátákba szúrjon egy nyilat és készítsen lenyomatot; és 2 mp-ig jelenítsen meg egy nevet a listából!
- Futtasd, és mentsd!



18. LECKE / LISTÁKBAN LÉVŐ SZÁMOK VIZSGÁLATA

19.) Logikai feladat:

Milyen számon parkol ez autó?

Megoldás: _____

Miért: _____

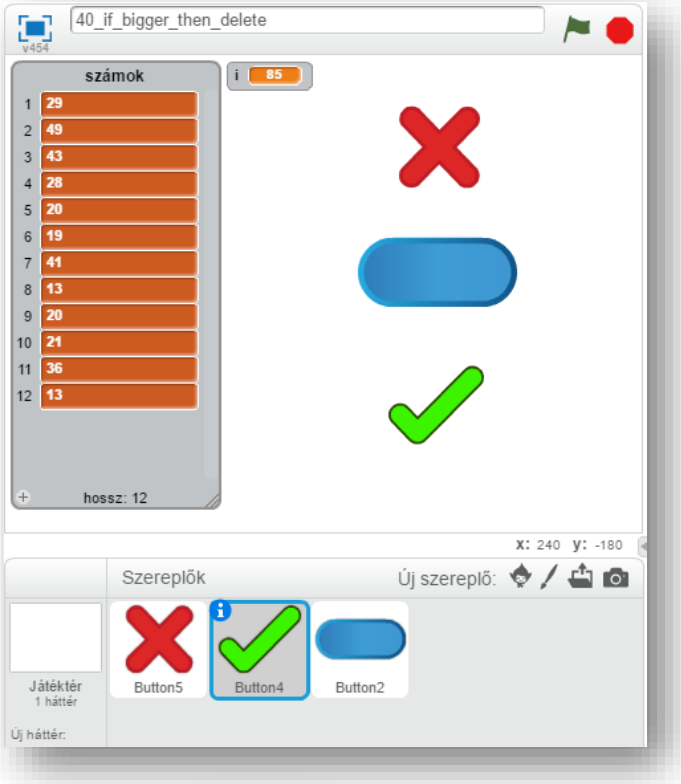
_____

22.) Gyakorlat:

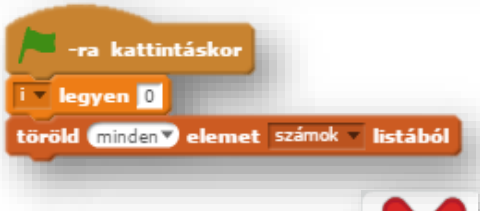
Ez talán az egyik legfontosabb programozási feladat! Egy adott listából bizonyos feltételeknek megfelelő számokat kitörölünk!

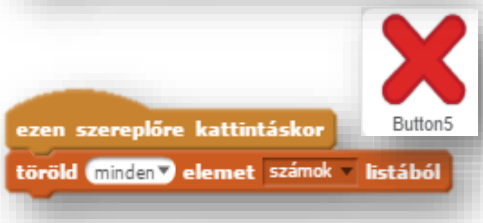
Most egy „számok” listába generálunk 10 darab véletlen számot 1 és 100 között! Majd a listából az 50-nél nagyobbakat kitöröljük.

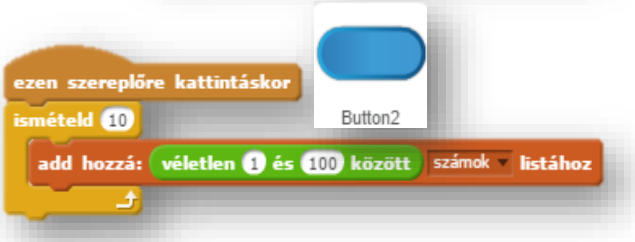
Először készítsük elő a programot! Nézzük meg mire lesz szükségünk ahhoz, hogy meg tudjuk oldani a feladatot!

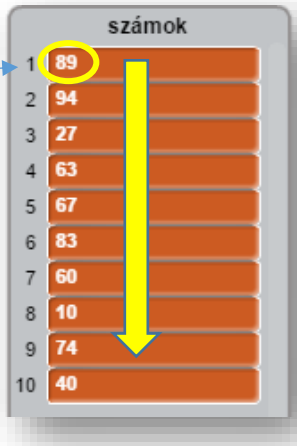


- A projekt neve legyen 40_if_bigger_then_delete!
- Vegyünk fel három szereplőt: Button5; Button4; Button2!
- Hozzunk létre egy „számok” nevű listát, és „i” nevű változót, melyre a későbbiekben szükségünk lesz! Ezzel az „i” nevű változóval fogunk lépkedi a listában!
- A program futása során végig látható a munkaterületen a változó értéke és a lista is!
- A programot zászlóra kattintással indítjuk! Ekkor törölni kell mindent, ami az esetleges előzőleg lefuttatott program során megmaradt! Az „i” változót lenullázzuk!
- A szereplőket, a listát, és a változót helyezzük el a minta szerint!
- A piros X gombra kattintáskor a program futása közben törölhetjük a lista tartalmát! Ezt programozzuk le!
- A kék gombra kattintással generáljunk 10 db véletlen számot 1 és 100 között!
- A zöld pipára való kattintással indul tulajdonképpen a programozási feladat!
- Nagyon fontos az „i” változó szerepe. Ennek a megértése a kulcsa mindennek!









- Az „i” értéke legyen 1! Ezzel indítjuk a feladatrészt!
- A lista minden elemének van egy sorszáma!
- Ezeken a sorszámokon megyünk végig egyesével, és vizsgáljuk meg, hogy megfelel-e a feltételnek!
- Indítunk egy ciklust, amelyet egyenlőre 100-szor futtatunk le!
- Melyben vizsgáljuk, hogy a lista „i”-edik eleme (jelen esetben 1) nagyobb-e mint 50! Mert ha igen, akkor töröljük ez az „i”-edik (jelen estben első elemét! Aztán az „i”-t csökkenteni kell 1-el, mert a ciklus következő futásánál megint 1-ről kell indulni, mert a törlés miatt a lista elejére egy másik szám került!
- Ha a lista első eleme nem volt nagyobb mint 50, akkor az „i”-t növelni kell eggyel, hogy a következő „kör” futásánál a 2. elemet vizsgálja!
- Így a következő körben vagy az 1-ről, vagy a lista 2. elemétől folytatja a vizsgálatot!
- Jelen példában a 89-et törli az első körben, ezért a lista visszacsúszik az elejére, és így a 89 lesz az első elem.
- Mivel a 89 is nagyobb mint 50, ezért ez is törölve lesz, és a lista megint előre csúszik!
- Ekkor a 27 lesz az első elem! Ez viszont már nem nagyobb mit 50, ezért ez megmarad a lista 1. elemének, tehát az „i”-t növeljük eggyel! Tehát a következő körben már a lista akkor második elemét vizsgálja, ami a 63.
- Mivel ez is nagyobb mint 50, ezért ezt törli, és visszacsúszik a következő elem a 67 a második helyre!
- Így folytatódik a program futása, míg el nem végzi a teljes lista vizsgálatát! Vagy ebben a programban, amíg az „i” el nem éri a 100-at!

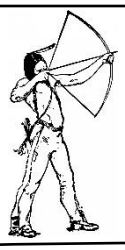
18.) Önálló feladat:

Ebben a feladatban generálunk 20 db véletlen számot 1 és 200 között! Majd a kiválasztott gombra kattintással megtartja a megfelelőket, a többit törli!

- A program neve legyen 41_odd_or_even_numbers!
- A gombokat, szereplőket készítsd el a minta lapján!
- Az előző programban tanultak alapján készítsd a programot!
- A megoldásban szükséged lesz a „mod” parancsra a páros és páratlanok kiválasztásánál! (szám mod 2 =1 ➔ pár.lan; szám mod 2 = 0 ➔ páros)
- Minden más kérdéses esetben szabadkezet kapsz!
- Végül mentsd a megadott helyre!

19. LECKE / A LISTÁBAN GENERÁLT SZÁMOK ÖSSZEGE

20.) Logikai feladat:



Ha két indián két perc alatt két nyilat lő ki, akkor tíz indián tíz perc alatt hány nyilat lő ki?

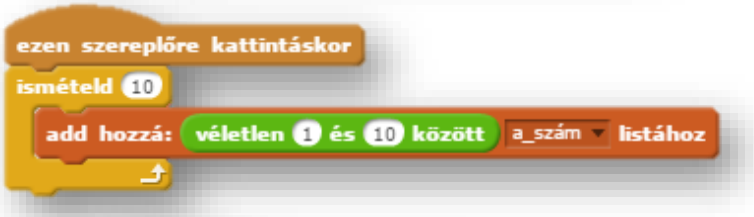
A megoldás: _____ db

Miért? _____

23.) Gyakorlat:

Ebben a feladatban egy gomb megnyomására generálunk 10 darab 1 és 10 közötti véletlen számot egy listába! Majd egy másik gomb megnyomására összeadja a listában szereplő számokat egy „összeg” nevű változóba!

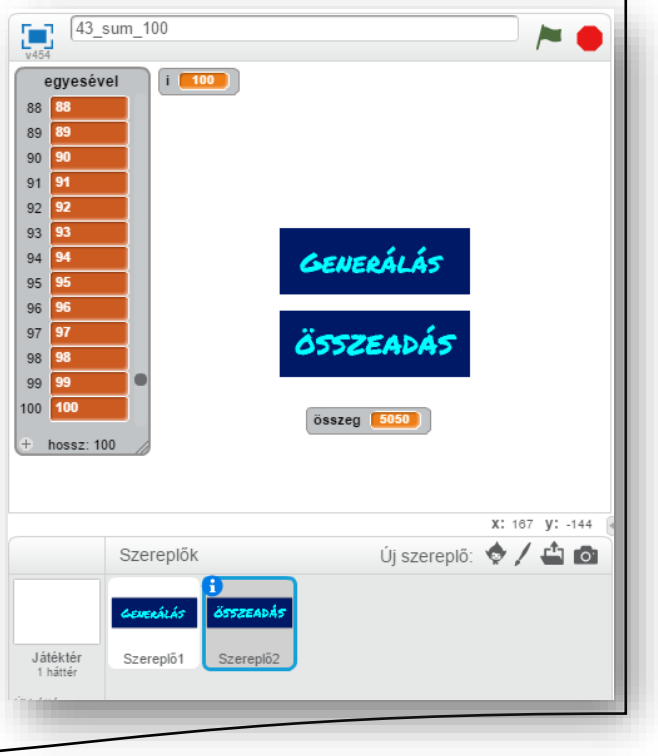
- A projekt neve legyen: 42_szum!
- Hozzunk létre egy „a_szám” nevű listát!
- Szükségünk lesz két változóra. Az egyik neve legyen „i”, mellyel a listában lépegetünk egyenként előre!
- A másik változó neve legyen „összeg”, melyben a listában található számok összegét tároljuk!
- Hozzunk létre két „nyomógombot”! Melyre készítsünk egy „Generálás” és „Összeadás” feliratot! Nevezzük el a szereplőket!
- A listát, a változókat, és a szereplőket helyezzük el a minta szerint!
- A program a zászlóra kattintással induljon!
- Nullázzuk le az „i” és az „összeg” változókat!
- Töröljünk minden számot a listából!
- A „Generálás” gombra kattintással hozzuk létre az „a_szám” listába 10 darab véletlen számot 1 és 10 között!
- Az „összeadás” gombra kattintva indítunk egy ciklust, melyet annyiszor futtatunk le, amilyen hosszú a lista (ahány eleme van aktuálisan)!
- Az „i” változót mindig léptetjük egyesével előre! Így jutunk a listában előre!
- Ahogy lépünk előre a listában az „összeg” változót egyenlővé tesszük az előző összeg érték és az aktuális szám összegével!
- Egy mp-et késleltessük a futásnál, hogy lássuk a lista és az összeg változásit!
- Végül mentsd a megadott helyre!



19.)Önálló feladat:

A feladat az első 100 szám összegének kiszámítása!

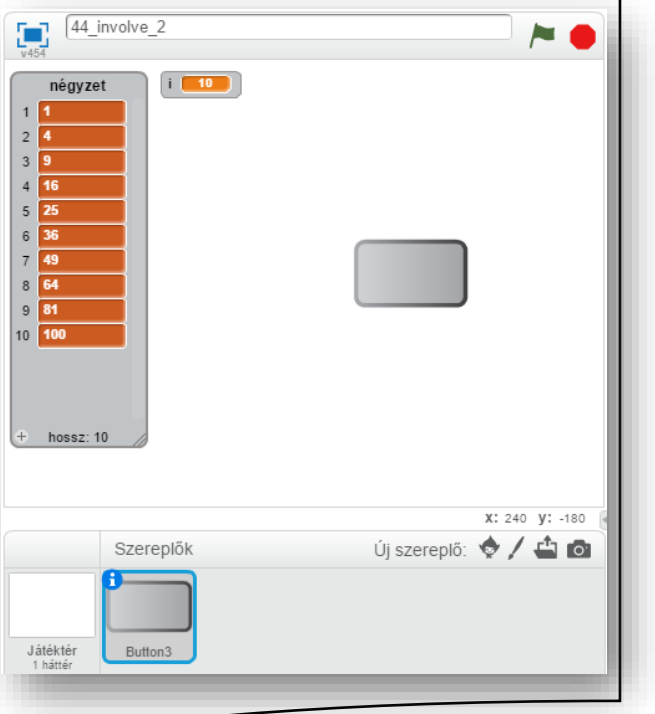
- A program neve legyen: 43_sum_100!
- Ezt a feladatot másképpen is meg lehetne oldani, de mivel a listákkal foglakozunk, ezért először generálunk egy „egyesével” nevű listába számokat egytől százig!
- Szükséged van két változóra is „i”-re és „összegre”!
- Még két gombot kell létrehoznod a minta szerint!
- Ciklusok készítésével, az előző feladatban tanultak alapján, és a minta szerint készítsd el önállóan a projektet!
- Végül mentsd a megadott helyre!



20.)Önálló feladat:

A feladat az első tíz szám négyzetét beletenni egy listába!

- A program neve legyen: 44_involve_2!
- Hozzál létre egy „négyzet” nevű listát!
- Szükség lesz egy „i” változóra!
- A szereplő közül válaszd a „Button3”-at!
- Készíts olyan ciklust, melyben az „i” változót eggyel növelve töltsd fel a listát a számok négyzetével!
- A lista feltöltése a szereplőre kattintással fusson le!
- Végül mentsd a megadott helyre!



20. LECKE / VÁLTOZÓK, ÉS LISTÁK ÖSSZEFOGLALÁS

21.) Logikai feladat:

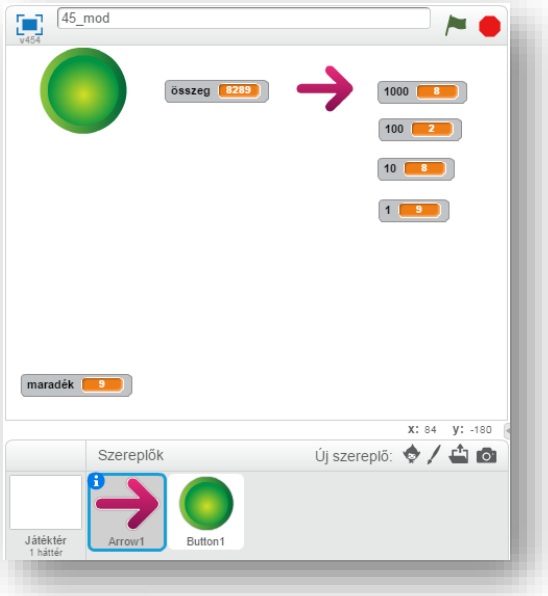
Az arányásó át akarja juttatni a folyó túl partjára az ott várakozó feleségének az aranyrögöt. A folyón dolgozik egy révész, aki minden mozdíthatót ellop. Az aranyásónak van egy ládikája, egy lakatja, és egy hozzátartozó kulcsa. És a feleségének is van egy lakatja egy hozzá tartozó kulccsal. A láda az egyetlen, ami túl nehéz ahhoz, hogy a révész ellopja. Segítsünk az aranyásónak átjuttatni az aranyrögöt anélkül, hogy ő vagy a felesége átkelne a folyón.

Megoldás röviden: _____

_____

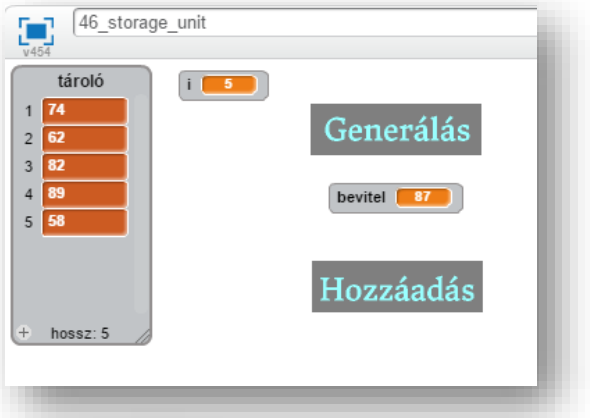
24.) Gyakorlat

A programban generálunk egy 1 és 1000 közötti véletlen számot, majd megnézzük, hogy hány darab ezres, százas tizes, és egyesből áll!



- A program neve legyen 45_mod!
- Hat változóra, és két szereplőre lesz szükségünk a feladat megoldásához!
- A Button1 és az Arrow1 szereplőket olvasd be, és hozd létre az „összeg”, „1000”, „100”, „10”, „1”, és a „maradék” nevű változókat!
- A program a zászlóra kattintással induljon, és nullázzuk le az összes változót!
- A „Button1”-el generáljuk a véletlen számot! Tehát az „összeg” változó legyen egyenlő a véletlen számmal!
- AZ „Arrow1”-re kattintva bontsa fel az „összeg” változóban lévő számot!
- Először azt kell megnézünk, hogy hány darab ezres van a generált számban! Ezt úgy tudjuk elérni, hogy elosztjuk az „összeg”-et 1000-rel. és a kapott számot lefelé kerekítjük egészre!
  Pl.: összeg: 6475 ➔ 6475/1000=6,475 ➔ lefelé ker.= 6
- Tehát ezt a számot beletesszük az „1000”-be!
- Viszont, hogy tovább tudjunk menni, ahhoz szükségünk van a „maradék”-ra, amit úgy kapunk meg, hogy a mod parancsot használjuk! Tehát a „maradék”-ot egyenlővé tesszük az „összeg” mod 1000-rel!
  pl.: maradék:= 6475 mod 1000 ➔ maradék=465
- Innentől fogva a logika ugyan az a változót egyenlővé tesszük az egész résszel, majd a maradékot kiszámoljuk, és azzal megyünk tovább!
  A legvégén, amikor már az „1”-es változónál megkaptuk a maradékot, ott csak egyszerűen beletesszük a változóba!
- Végül teszteld, és mentsd a megadott helyre!

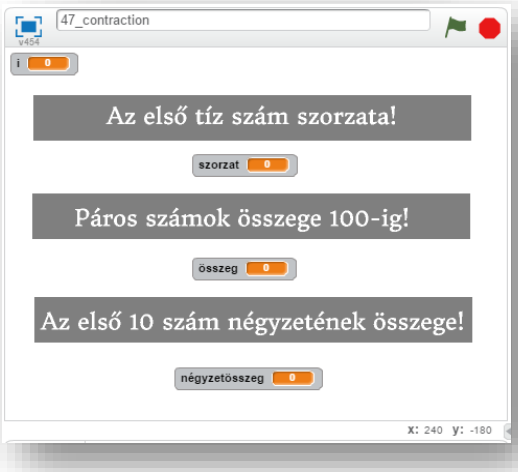
25.) Gyakorlat



Ebben a feladatban egy változóba kell generálni egy véletlen számot 1 és 100 között, majd ha úgy gondoljuk (mondjuk páros, vagy páratlan, vagy prím szám), akkor hozzáadjuk, ha nem, akkor generálunk egy másikat!

- A program neve legyen: 46_storage_unit!
- Hozzuk létre a szükséges változókat és listát!
- A lista neve legyen „tároló”!
- Kell egy „bevitel” nevű változó, amibe generáljuk a véletlen számot!
- Szükségünk lesz a szokásos „i” változóra, melyben a lista aktuális helyét tudjuk léptetni!
- Két szereplőre is szükségünk lesz, melyek nyomógombok! A minta szerint!
- A program zászlóra kattintással induljon!
- Ürítsük ki a listát, és nullázzuk le az összes változót!
- A „Generálás” szereplőre kattintáskor tegyünk a bevitel változóba egy és száz közötti véletlen számot!
- A „Hozzáadás” szereplőre kattintáskor növeljük az „i” értékét egyel, azért, hogy utána a „tároló” lista „i”-dik helyére beszúrhasuk a „bevitel” változóban lávő számot!
- (Ennek a feladatrésznek van egy egyszerűbb megoldása is, de a későbbi programok miatt itt így oldjuk ezt meg! Egyébként „i” nélkül a „Hozzáadás” gombra kattintáskor csak ezt az egy sort kell megadni!)

21.)Önálló feladat:



Ebben a feladatban három részfeladatot kell megoldani!

- A program neve legyen: 47_contraction!
- A három „gomb” szereplőt, és a hozzá szükséges változókat hozd létre a minta szerint!
- A változók legyenek: i, szorzat, összeg, és négyzetösszeg!
- A gombokat lásd el a feliratokkal, és írd meg a hozzájuk tartozó algoritmust!
- Minden egyéb kérdésben szabadkezet kapsz!
- Ha végeztél mentsd a megadott helyre a programfájlt!

Ismertető, összegző: Egy program megírásának lépései:

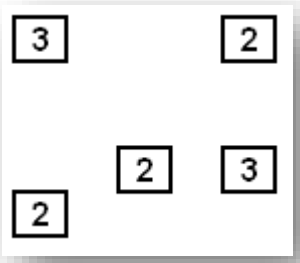
- Először elolvasva a feladatot értelmezzük, és végiggondoljuk, hogy tulajdonképpen mi is a dolgunk!
- Létrehozzuk a szükséges változókat, listákat, szereplőket, háttereket! (A későbbiek során bővíthetjük, ha szükséges!)
- Logikusan, lépésről lépésre, az elejétől feladatokat adunk a szereplőknek, változókat, listákat felhasználva algoritmusokat, ciklusokat készítünk!
- A program írása során teszteljük a részprogramokat, javítjuk az esetleges hibákat, mentünk!

21. LECKE / EGYMÁSBA ÁGYAZOTT CIKLUSOK

22.) Logikai feladat:

Ez egy nagyon egyszerű „Kösd össze, ha tudod!” feladat!

A lényeg az, hogy csak vízszintes és függőleges vonalak húzásával létesíts kapcsolatot a szomszédos téglalapokkal! Csak annyi vonal indulhat az alakzatból, ahányas szám van a téglalapban!



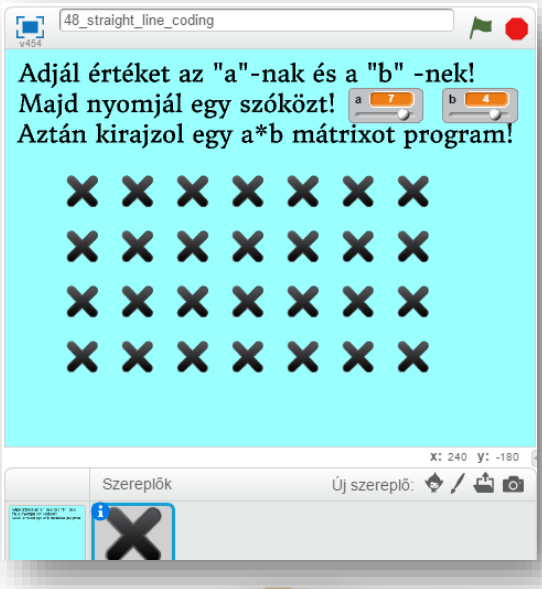
26.) Gyakorlat:

Az egymásba ágyazott ciklusok készítése, szintén az alap programozási feladatok közé tartozik!

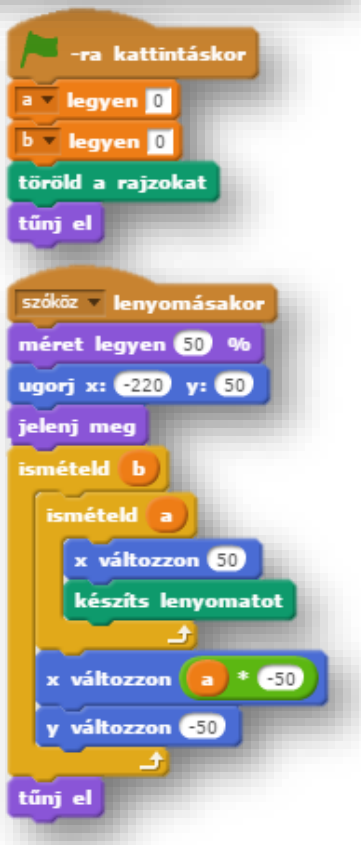
Ilyenkor lesz egy külső és egy belső ciklusunk!

Nézzük a magyarázatot egy példán keresztül:

A feladat egy „a” és egy „b” változó bekérése után, kirajzolni egy a*b mátrixot! Tehát, ha például az a=7 és b=4, akkor kirajzol egy 7 oszlopból és 4 sorból alakzatot!



- A program neve legyen: 48_straight_line_coding!
- A feladat megoldásához szükség lesz két változóra, és egy szereplőre!
- Az „a” változó legyen csúszka típusú, és legyen a min. értéke 0, max. értéke 8!
- A „b” változó legyen szintén csúszka típusú, és legyen a min. értéke 0, max. értéke 5!
- A hátteret készítsd el a minta alapján, és írd rá a szöveget! A változókat tedd az üres helyre!
- Vedd fel a „Button5” szereplőt, melyet majd 50%-os nagyításban használunk!
- A program a zászlóra kattintással induljon! Nullázzuk le a változókat, töröljük az előző lenyomatokat! A szereplőt az elején tegyük láthatatlanná!
- Miután megadtuk a változókat, szóköz lenyomására kezdje a kirajzolást!
- A lenyomatokat fogunk készíteni, az elsőnek a helyét adjuk meg az elején! (-220,50)
- Jelenjen meg a szereplő, amit az elején elrejtettünk!
- A belső ciklust úgy építsük fel, hogy az „x” koordinátát változtatjuk 50 képponttal, és lenyomatot készítünk a szereplőről „a” darabszor! Ez fogja a sorokat kirajzolni!
- A külső ciklusban tulajdonképpen a sortörés készítjük el, „a”*-50 távolságot visszalép a szereplő „x” irányba, és -50 kp lép a szereplő lefelé „y” irányba! Ezzel lépünk egy sornyit lejjebb és vissza a sor elejére! Ezt „b”-szer ismételjük!
- A végén eltüntetjük a szereplőt!
- Teszteljük, mentjük!



27.) Gyakorlat:

Ebben a programban tegyük fel, hogy egy elektronikai boltban egy hét alatt eladott TV-k darabszámát ábrázoljuk „oszlopdiagramon”!

Az alaphátter a „kép_napok”, csak be kell olvasni!

Szükségünk lesz egy hét elemű listára, melybe véletlen számokat generálunk 1 és 10 között!

Kell két szereplő, a „Generál” és a „Kirajzol” gombok! Melyeket mi készítünk el!

Végül létrehozunk egy kis kék négyzet szereplőt, melyet az oszlopok „kirajzolásakor” használunk fel!

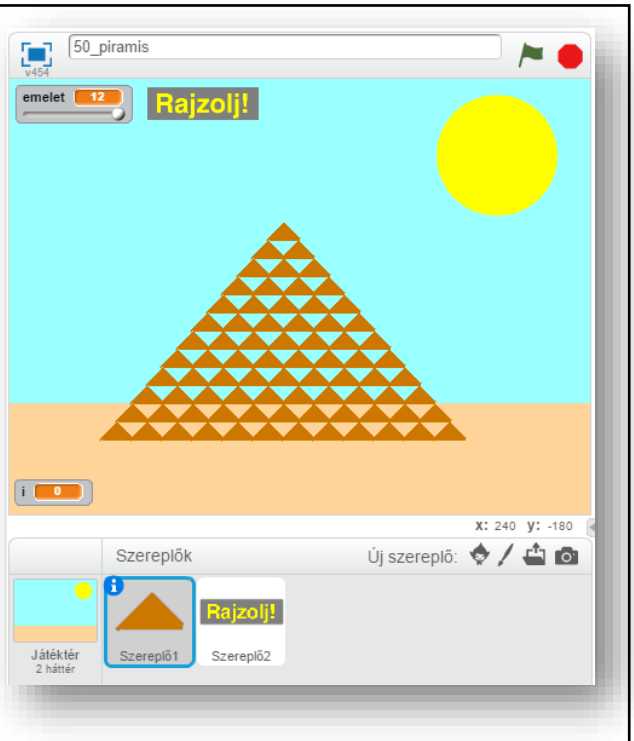
Szükségünk van még két változóra, „i” -re és „szám”-ra! Az „i” –vel haladunk 1-7-ig a hét napjain, egyben a lista sorszámain! Erre a külső ciklusban lesz szükségünk!

A „szám” változóba pedig a lista „i”-dik helyén álló szám kerül! Mellyel ciklusban az „x” koordináta változtatásával, és lenyomat készítésével „rajzoljuk” meg az oszlopokat!

- A program neve legyen: 49_sale_data!
- Olvassuk be a hátteret, hozzuk létre a listát, és a változókat! Készítsük el a szereplőket!

- A program a szokásos módon indul! Nullázzunk, töröljük az előzőeket!
- A „Generál” gombra kattintva létrehozzuk az ábrázolandó adatokat!
- A „Kirajzol” gombra kattintva egy „rajzolj” üzenetet küld!
- A kék négyzet szereplő a „rajzolj” üzenet érkezésekor kezdi el a „főprogram” futtatását!
- Az eddigiekkel ellentétben, most az elkészített programon keresztül értsük meg az utasítások miértjét! Nézzük meg, mi után mi következik!
- Majd önállóan készítsük el a programot, ha lehet segítség nélkül!
- A kész programot mentsük a megadott helyre!

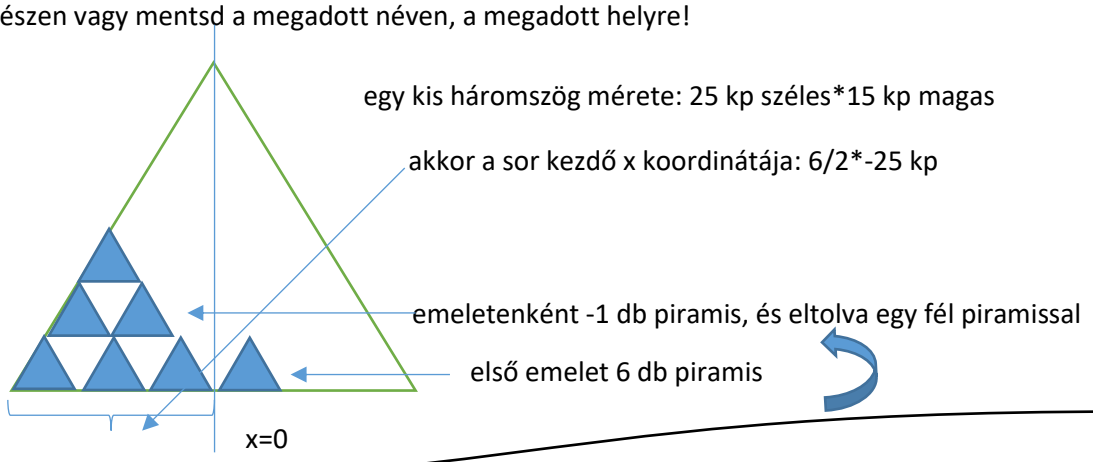
22.)Önálló feladat:



Ebben a programban egy piramist kell „építened”! Egy csúszkával add meg, egy változóban, hogy milyen magas, hány emeletes legyen az épület!

- A program neve legyen: 50_piramis!
- Rajzolj egy hátteret, melyben kitöltött téglalapokkal készítsd el az eget, és a sivatag homokját! Rajzolj egy sárga kitöltött napot a jobb felső sarokba, a minta szerint!
- Szükség lesz két szereplőre! Az egyik egy „Rajzolj!” gomb, melyre kattintva kirajzolódik majd a piramis!
- A másik pedig egy kis barna háromszög, melynek lenyomataival rajzoljuk ki a teljes piramist! A kis háromszög mérete legyen 25*15 képpont, hogy könnyen tudjunk számolni a koordinátákkal!
- Kelleni fog két változó is! Az egyik egy „emelet” nevű, melyet csúszkává alakítunk! A minimumát nullára, a maximumát tizenkettőre állítsd! Ezzel állítod be az emeletek számát!
- A másik változó legyen egy „i”, melyben majd csökkenteni fogjuk emeletenként a kirajzolt kis piramisok számát!
- A program a zászlóra kattintással induljon, és mindent nullázzunk le, a kis piramist tegyük láthatatlanná! Töröljük az előző rajzokat!
- A „Rajzolj!” gombra küldjön a program egy „építs” üzenetet!
- A kis piramis szereplő, ha megkapja az „építs” üzenetet, akkor kezdje építeni az emeletes piramist!
- A rajzolást középen kezdje, az emeltek számától függően! Tehát ha 1 emelet megadva, akkor középre készítsen lenyomatot egy darab kis piramist! Ha 2 emelet van megadva, akkor arányosan ballra kezdjen rajzolni a közepéhez képest, és így tovább! Minél több emelet számot adunk meg, annál messzebbről kell kezdeni a rajzolást a középvonalhoz képest, a bal oldalon!
- Amikor egy emelettel feljebb megyünk, akkor csökkenteni kell a kirajzolandó piramisok számát, és egy fél piramisnyival kell beljebbről indulni!
- Közben meg kell jeleníteni a kis piramis, és el kell tüntetni! A megfelelő helyen pedig lenyomatot készíteni!
- A programot egymásba ágyazott ciklusokkal kell elkészítened!
- Ha készen vagy mentsd a megadott néven, a megadott helyre!

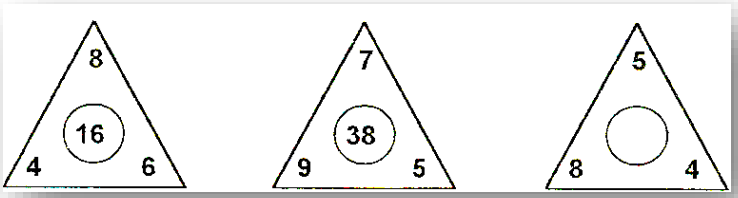
Pl.: 6 emelet



22. LECKE / A LEGKISEBB, ÉS LEGNAGYOBB ELEM KIKERESÉSE EGY LISTÁBÓL

23.) Logikai feladat:

A háromszögekben lévő számok felhasználásával jöjjél rá a szabályra (műveletre)! Írd be a harmadik körbe a jó megoldást!

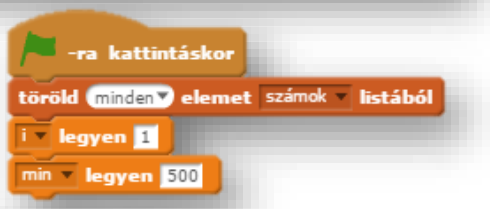


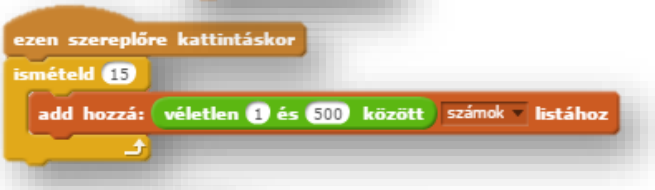
Egy lista (tömb) legkisebb és/vagy legnagyobb elemének kiválasztásához bevezetünk két változót, a min-t és max-ot. Kezdetben beállítjuk mindkét változó értékét a tömb első elemének értékére - feltételezve hogy ez a legkisebb és legnagyobb elem is a tömbben van, majd a második elemtől a tömb végéig végignézzük az elemeket (van-e a beállított első elemnél nagyobb vagy kisebb). Ha bármelyik elem kisebb mint a min változó értéke, akkor az új elem értékét megjegyezzük a min változóban. Hasonlóan, ha olyan elemet találunk, amely nagyobb mint a max értéke, akkor azt megjegyezzük a max változóban. Így a ciklus lefutása (tömb elemeinek átnézése) után a min és a max változók a tömb legkisebb ill. a legnagyobb elemét fogja tartalmazni.

28.) Gyakorlat:

Ebben a feladatban egy 15 elemű listából (tömbből) kell kiválasztani a legkisebb elemet!

- A program neve legyen: 51_minimum!
- Szükségünk lesz egy „számok” nevű listára, és két változóra! A változók neve legyen „i” és „min”!
- A megoldáshoz két kék nyomógombot (szereplőt) készítünk! Az egyiken „Generálj!” felirat, a másikon „A legkisebb!” felirat szerepeljen!
- A programot a zöld zászlóra kattintással indítjuk! Törlünk mindent a „számok” listából. és lenullázzuk az „i”, a „max”-ot pedig 500-ra állíjuk (ez a legnagyobb szám, ami tárolható a listában)!
- A „Generálj!” gombra kattintva hozzunk létre 15 darab, 1 és 500 közötti véletlen számot a „számok” listába!
- A listában a megszokott módon haladunk az „i” növelésével, egyesével előre!
- A „Legkisebb!” gombra kattintásnál a lista első elemét beletesszük a „min”-be, és indítjuk a ciklust amit 14 szer futtatunk le, mert így ér végig a listában. Majd „i”-vel haladunk egyesével, és ha paranccsal vizsgáljuk, hogy nagyobb-e a lista aktuális eleme, mint a „min”-ben tárolt eddigi legkisebb elem.
- A végén, mentsd a magadott helyre a programot!





23.)Önálló feladat:

Először kell olyat csinálnod, hogy egy előzőekben megírt programot kell bővítened, folytatnod!

- Nyisd meg az 51_minimum nevű programot, és mentsd más néven 52_basic_math néven!
- Három új szereplőt kell létrehoznod a mint szerint, piros téglalapba a következő szöveggel! („A legnagyobb!”; „Az átlag!”; és „Az összeg!”)
- Új változókat is kell létre hoznod: „max”; „átlag”; „összeg” néven! Ezeket a változókat a program indításánál le kell nulláznod!
- Mivel az „i”-t többször fogod felhasználni, ezért, minden részprogram végén le kell nulláznod! (A legkisebb résznél is be kell tenned azt a parancsot, mellyel az „i” értékének 0-t adsz!)
- Egyértelműen az a feladat, hogy a „max”-nál a legnagyobb számot kell kiválasztani a listából! Tulajdonképpen a „legkisebb” részprogramot kell duplikálni, és az előjelet megváltoztatni az összehasonlításnál. Majd a „min” változót „max”-ra cserélni!
- Az „átlag” és az” összeg feladatot összekötheted, mert az átlag kiszámításánál úgyis szükséged van az „összegre”!
- „átlag”=”összeg”/15 (Ha szükséged van esetleg plusz változóra, akkor felvehetsz egyet!)
- Futtasd, és teszteld a programot, majd mentsd a megadott helyre!

24.)Önálló feladat:

Ismét az előző feladatot kell bővítened, illetve megváltoztatnod!

Azt a problémát kell megoldanod, hogy ha a program újraindítása nélkül nyomod meg a „Generálás!” gombot, akkor plusz 15 elemet (számot) hozzáad a listához! Így 30 vagy 45, illetve több eleme lesz a tömbnek! De csak az első 15-őt vizsgálja!

Ezért az a feladatot (teljes programot), változtasd meg a úgy, hogy bármekkora listánál is működőképes legyen a program!

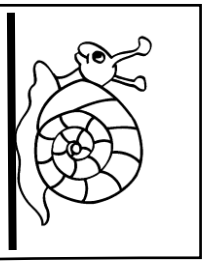
- A program nevét változtasd meg: 53_full_math_program-ra!
- Az előzőekben leírtakat úgy tudod megoldani, hogy az ismétlések számát (14) a „tömb hossza - 1”-re állítod!
- Minden más előforduló kérdésben szabadkezet kapsz!
- Futtasd, és teszteld a programot, majd mentsd a megadott helyre!

23. LECKE / EGY LISTA ELEMEINEK NÖVEKVŐ SORRENBE TÉTELE, EGYSZERŰ CSERÉS RENDEZÉSSEL

24.) Logikai feladat:

Egy csiga beleesett egy 10 méter mély kútba. Minden nap 2 métert mászott felfelé, éjjel 1 métert visszacsúszott. Hány nap alatt jut ki a kútból?

A megoldás: _____



A feladatban egy N elemű listát nagyság szerint sorba kell rendezni. Nagyon sok rendezési algoritmus létezik, ezek közül az egyik az egyszerű cserés rendezés. Egy rendezetlen sorozaton futtatott algoritmus eredménye a rendezett sorozat. Amely helyben keletkezik, így az eredeti sorrend elvész. Az a rendezési algoritmus jó, amelynek kicsi a tárigénye és nagyon gyorsan végrehajtja a rendezést, és persze egyszerű, könnyen megérthető a működése.

A listát nevezzük „a” –nak! Szövegben jelöljük a() –val! A lista első elemét: a(1)-nek, a másodikat a(2)-nek, és így tovább! Egy „n” elemű tömböt: a(n)-el írhatunk le!

Hasonlítsuk össze a lista első elemét a sorozat összes többi mögötte lévő elemével, s ha valamelyik kisebb nála, akkor cseréljük meg azzal! Ezzel elérhetjük, hogy a sorozat első helyére a legkisebb elem kerül. Folytassuk ugyanezen elven a sorozat második elemével, utoljára pedig az utolsó előttivel.

Az i-t az első elemre állítjuk, a j-t a másodikra! Tehát vizsgáljuk, hogy az a(2) kisebb-e mint az a(1). Ha igen, akkor cseréljük őket! Haladni a lista elemein egymásba ágyazott ciklussal tudunk. A külső ciklusban az „i”-vel, a belső ciklusban a „j”-vel haladunk!

A j -vel végigértünk a sorozaton, és az 1. helyre megtaláltuk a sorozat legkisebb elemét. Most a(1) biztosan a legkisebb elem. Az i -t növeljük eggyel és a 2. helyre keressük meg, a j segítségével, a(2), a(3), a(4), a(5) közül a legkisebbet. Ez persze biztos nem lesz kisebb a(1) - nél, különben ő került volna az 1. helyre. A fentiekben leírt gondolatmenetet követjük, amíg i -vel végig nem érünk a sorozaton, azaz az n-1. helyre meg nem találtuk a(n-1) és a(n) közül a kisebbet. Az n. helyre pedig nyilvánvalóan a sorozat legnagyobb eleme marad. Ebben az egyszerű példában az n=5, mivel a lista csak öt elemből áll!

Alap:

	1	2	3	4	5
rendezetlen	**5**	**3**	**9**	**1**	**7**

Végeredmény:

	1	2	3	4	5
rendezett	**1**	**3**	**5**	**7**	**9**

1. lépés:

	1	2	3	4	5
	i ↓	j ↓			
csere	**5**	**3**	**9**	**1**	**7**
	i ↓		j ↓		
nincs csere	**3**	**5**	**9**	**1**	**7**
	i ↓			j ↓	
csere	**3**	**5**	**9**	**1**	**7**
	i ↓				j ↓
nincs csere	**1**	**5**	**9**	**3**	**7**

2. lépés:

	1	2	3	4	5
		i ↓	j ↓		
nincs csere	**1**	**5**	**9**	**3**	**7**
		i ↓		j ↓	
csere	**1**	**5**	**9**	**3**	**7**
		i ↓			j ↓
nincs csere	**1**	**3**	**9**	**5**	**7**

3. lépés:

	1	2	3	4	5
			i ↓	j ↓	
csere	**1**	**3**	**9**	**5**	**7**

és így tovább…

29.) Gyakorlat:

Tehát készítsünk egy programot, melyben gombnyomásra generálunk egy „a” nevű, 10 elemű listába, véletlen számokat 1-100-ig, és egy másik gomb megnyomására rendezze sorba a lista elemeit! Használjuk a mintán látható változóneveket!

- A program neve legyen: 54_grade!

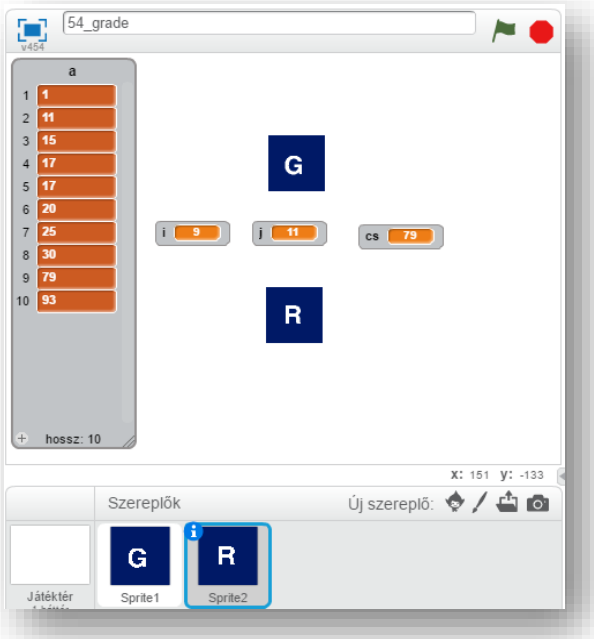
(A megoldás a következő oldalon található, de próbáljuk először megoldani a feladatot anélkül, hogy fordítanánk, és megnéznénk!)

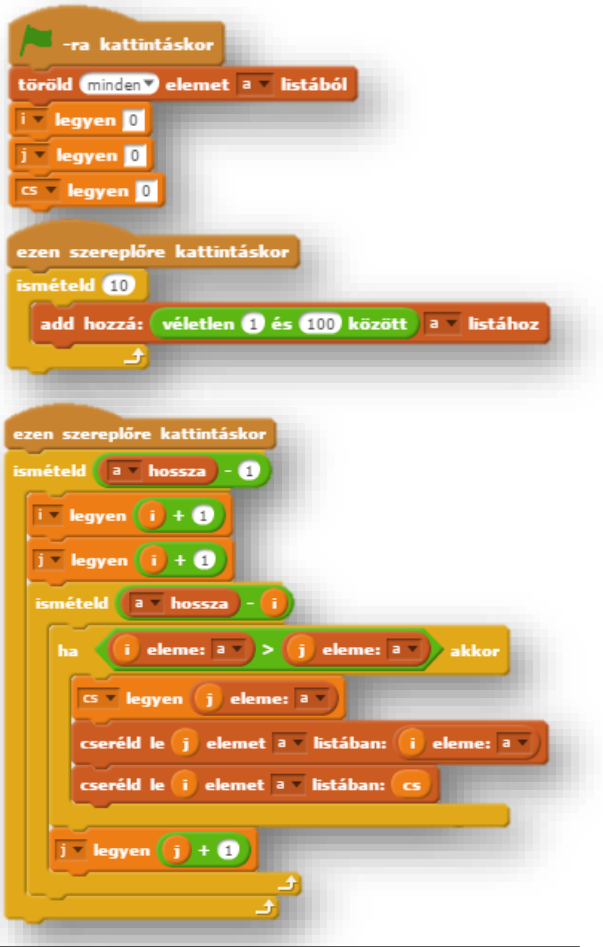
Egyszerű cserés rendezés megoldása:

A lista neve: a

A változók: i; j; cs;

Két szereplő rajzolva: G; R;

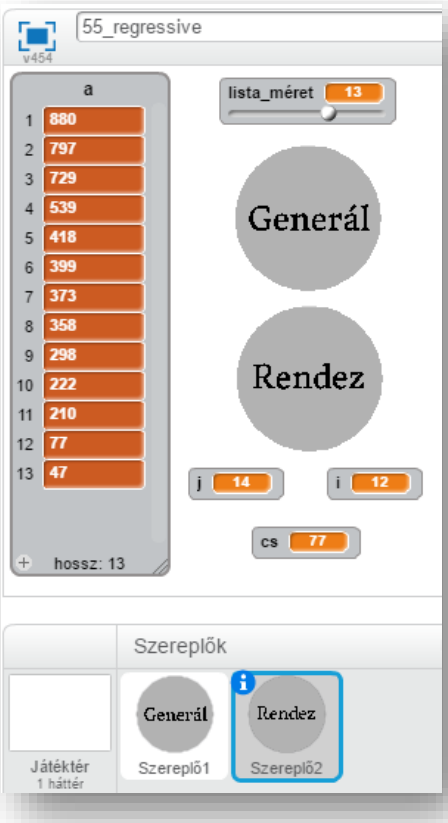




25. Önálló feladat:

Ebben a feladatban a program indítása után egy „csúszkára” állított változóban meg kell adni, hogy mekkora méretű listát szeretnénk majd létrehozni. Aztán egy kör alakú gombra kattintva generáljon változó számú, 1 és 1000 közötti számot. Majd a listát csökkenő rendbe rendezni, egy „Rendez” gomb megnyomásával!

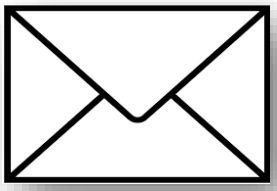
- A program neve legyen: 55_regressive!
- Készítsd elő a programot listával, változókkal, szereplőket! (a; lista_méret; i; j; cs;”Generál”; „Rendez”)
- A minta és a leírás alapján készítsd el a programot!
- A csúszkát úgy állítsd be, hogy 1 és 20 közötti számot lehessen bevinni!
- Futtasd, és teszteld a programot, majd mentsd a megadott helyre!



24. LECKE / EGY LISTA ELEMEINEK NÖVEKVŐ SORRENBE TÉTELE, MINIMUMKIVÁLASZTÁSOS RENDEZÉSSEL

25.) Logikai feladat:

Két borítékot és két képeslapot vettem. Egy boríték 5 Ft-tal olcsóbb, mint egy képeslap. 50 Ft-ot fizettem. Mennyibe került egy képeslap, mennyibe egy boríték?



Képeslap: _____ Ft Boríték: _____ Ft

Az előző módszer hátránya a sok felesleges csere. Célszerűbb lenne az aktuális, i. , elemet a mögötte lévők közül egyedül a legkisebbel felcserélni. Ez a felismerés vezet a módszer javításához, a minimumkiválasztásos rendezéshez.

A minimunkiválasztásos rendezés lépései:

Az a() tömb 1. helyére keressük ki az a(1), a(2), a(3), a(4), a(5) elemek közül a legkisebbet (minimumkiválasztás)! Ha megtaláltuk a legkisebb elemet, akkor cseréljük ki az a(1) - gyel!

Ekkor az a(1) biztosan a vektor legkisebb eleme. Növeljük az i -t 1 -gyel, így az a() tömb 2. helyére keressük ki a(2), a(3), a(4), a(5) közül a legkisebbet!

Most a(i) -t cseréljük fel a(min) -nel! (Ez a csere tulajdonképpen felesleges, hiszen most a(2) - t cseréljük fel a(2) -vel! növeljük i -t 1 -gyel!

Most a(i) -t cseréljük fel a(min) -nel! majd növeljük i -t 1 -gyel!

Most a(i) -t cseréljük fel a(min) -nel! az i -vel elértük n-1 -t, ezután az a() biztosan rendezett!

A módszer hátránya, hogy bizonyos esetekben (ha i=min) egy elemet önmagával cserélünk fel. Ez nyilvánvalóan felesleges csere.

Alap:	1	2	3	4	5
rendezetlen	**5**	**3**	**9**	**1**	**7**

Végeredmény:	1	2	3	4	5
rendezett	**1**	**3**	**5**	**7**	**9**

Lépések:

	i	M			
csere	**5**	**3**	**9**	**1**	**7**

	i			M	
csere	**5**	**3**	**9**	**1**	**7**

	i	M			
magával csere	**1**	**3**	**9**	**5**	**7**

			i	M	
csere	**1**	**3**	**9**	**5**	**7**

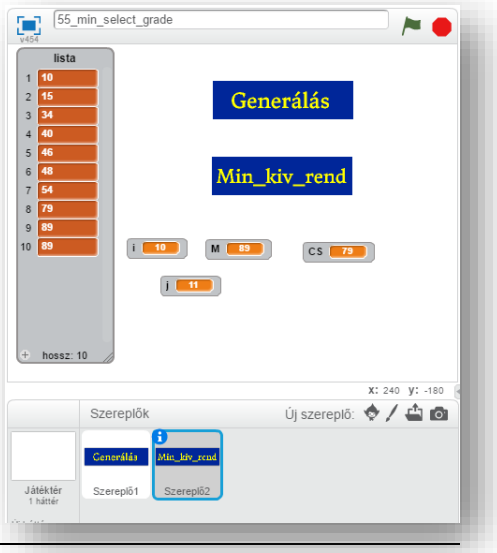
				i	M
csere	**1**	**3**	**5**	**9**	**7**

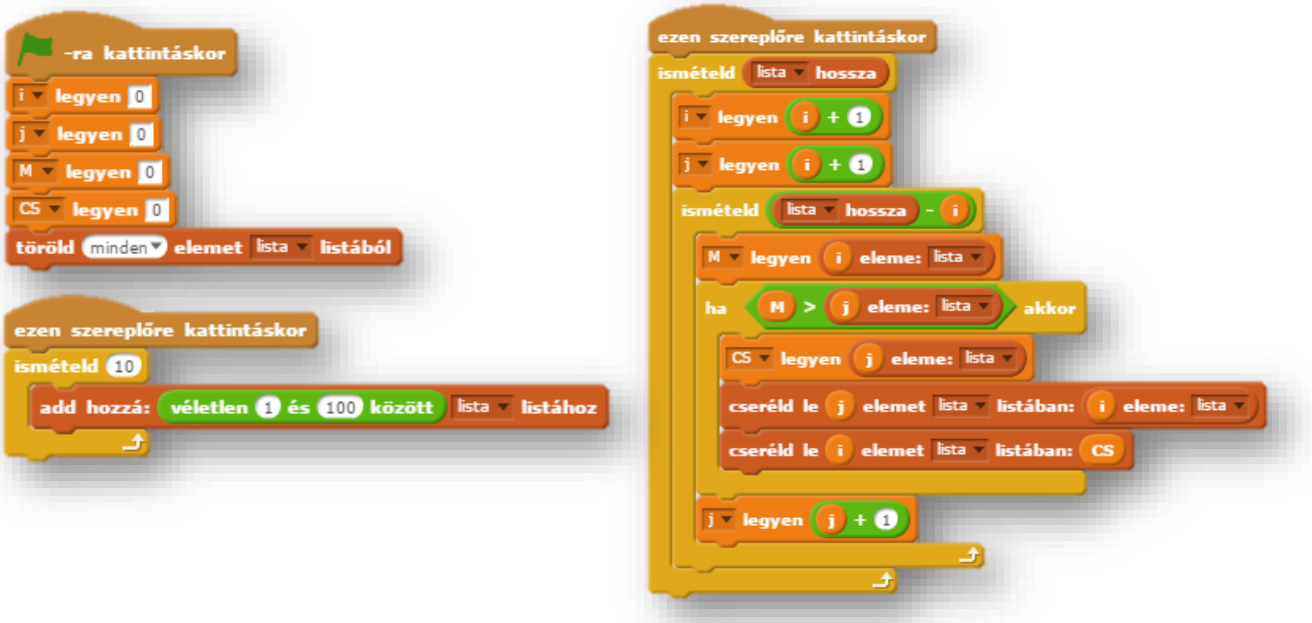
				i	M
	1	**3**	**5**	**7**	**9**

30.) Gyakorlat:

Készítsünk az előző algoritmus felhasználásával egy programot, melyben egy 10 elemű tömb elemeit rendezzük növekvő sorrendbe minimumkiválasztásos rendezéssel!

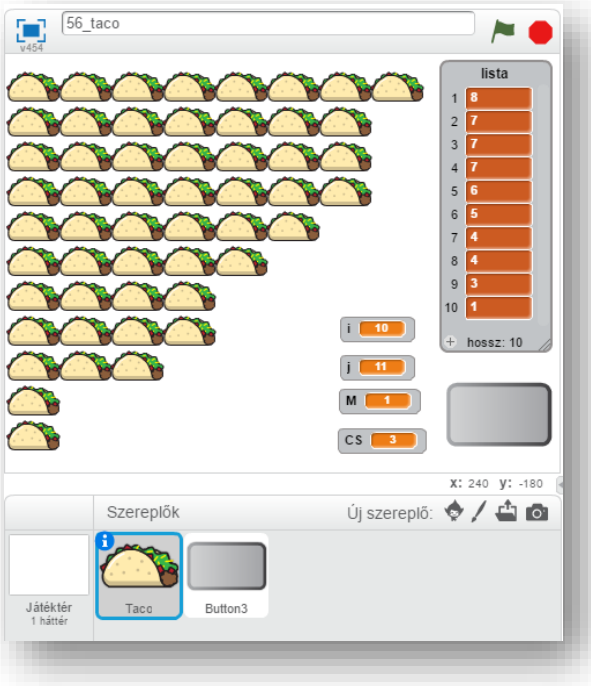
- A program neve legyen: 55_min_select_grade!
- A lista neve legyen: lista! A programhoz szükséges változók legyenek: i; j; M (minimum); CS (csere)!
- Készíts két szereplőt gombnak a minta szerint!
- Futtasd, és teszteld a programot, majd mentsd a megadott helyre!





26.) Önálló feladat:

Ebben a programban egy „taco-evő verseny” eredményit kell ábrázolni! A versenyen 10 ember indult, és 10 taconál senki nem tud többet enni!



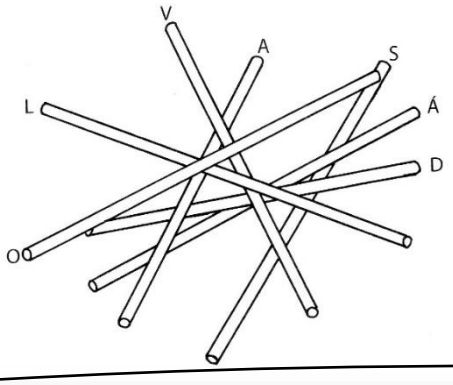
- A program neve legyen 56_taco!
- Először generálj egy listába véletlenszerűen számokat az előzőekben leírtak szerint!
- Majd a listában szereplő számokat rendezd minimumkiválasztásos rendezéssel csökkenő sorrendbe!
- Az eddigieket, mind egy gomb lenyomására tegye meg!
- A rendezett lista alapján rajzolja ki, hogy az első helyezett hány darab taco-t tudott megenni!
- Majd alá, a második, harmadik, és így tovább a többi helyezett teljesítményét!
- A programot a mintakép alapján készítsd el, de a saját elképzelésed szerint programozd le!
- A lista, a változók, a szereplők nevének megadásában is szabadkezet kapsz!
- A program futtatása alatt a lista és a változók legyenek elrejtve!
- Futtasd, és teszteld a programot, majd mentsd a megadott helyre!

25. LECKE /TÖBB LISTA HASZNÁLATA, MŰVELETEK LISTÁKKAL, SZÉTVÁLOGATÁS

27.) Logikai feladat:

Milyen sorrendben kell levenni a pálcákat egymásról? Írd a betűk sorrendjét a vonalra!

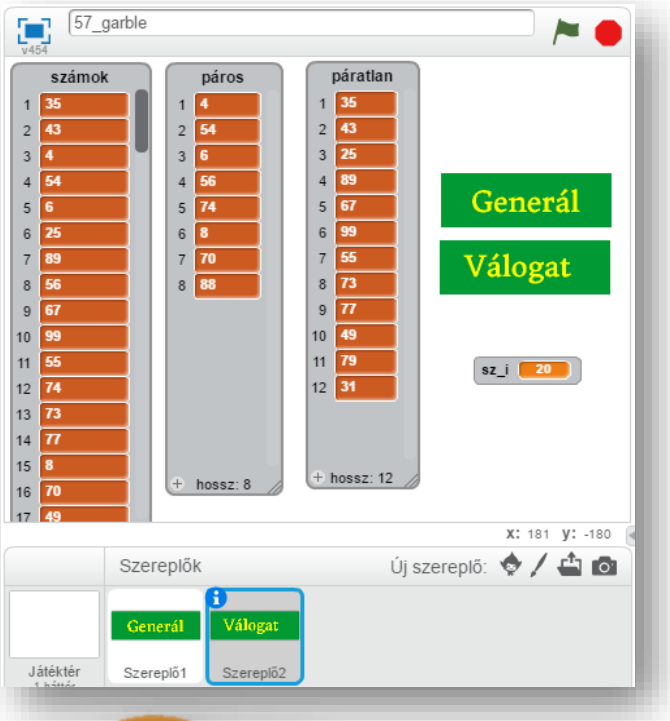
Megoldás: _____

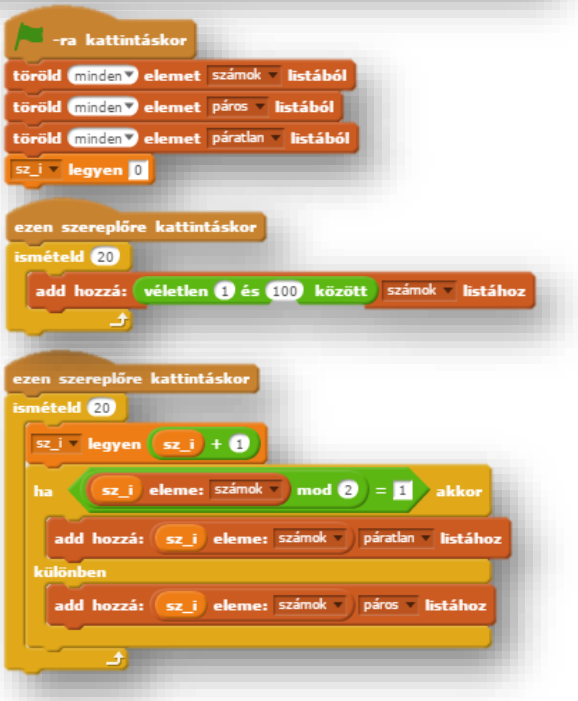


31.) Gyakorlati feladat:

Ebben a feladatban egy 20 elemű listát fogunk szétválogatni! A páros és a páratlan számokat esszük bele két külön listába!

- A program neve legyen 57_garble!
- Hozunk létre három listát: „számok”; „páros”; és „páratlan” néven!
- Szükségünk lesz a „számok” listában való lépkedéshez egy: „sz_i” nevű változóra!
- Két új szereplőt kell létrehoznunk a nyomógomboknak a minta szerint!
- Az elsővel generálunk a „számok” listába 20 darab véletlen számot 1 és 100 között!
- A program szokás szerint a zászlóra kattintással induljon, és töröljünk minden eddig használt listát és változót!
- A „Válogat” gombra kattintással fusson le egy olyan algoritmus, melyben végigmegy mind a 20 tárolt elemen, és megvizsgálja egyenként, „mod” parancs segítségével, hogy az adott elem páros-e vagy páratlan!
- Erre a „Ha, akkor, különben” parancsra lesz szükségünk! Ha a „számok” i. eleme osztva kettővel egyet ad maradékul (számok(i) mod 2 = 1), akkor adja hozzá a „páratlan” listához, különben adja hozzá a „páros” listához!
- Futtasd, és teszteld a programot, majd mentsd a megadott helyre!





27.) Önálló feladat:

Készíts egy programot az előzőhöz hasonlóan, melyben egy listába generálsz 30 db 1 és 500 közötti számot! Majd három másik listába kigyűjtöd a 3-al, 5-el, és 8-al osztható számokat! Használj „Generál”, és „Válogat” gombokat!

- A program neve legyen: 58_sort!
- Mindenben szabadkezet kapsz! Mentsd a megadott helyre!

26. LECKE /TÖBB LISTA HASZNÁLATA, MŰVELETEK LISTÁKKAL, ÖSSZEFÉSÜLÉS

27.) Logikai feladat:

Mennyi ideig isznak a matrózok?

Ha hat jókedvű, pörge bajszú, nagy pocakú matróz hat kupica pálinkát egy perc alatt tud legurítani, akkor 12 jókedvű, pörge bajszú, nagy pocakú matróz kétszer annyi kupica pálinkát mennyi idő alatt tud legurítani?

Megoldás: _____

Két rendezett tömböt, „a”-t és „b”-t összefésülünk egy harmadik „c” nevű tömbbe! Az a és b tömbök a program elején adottak, melyek elemeiből a program hozza létre a c tömböt.

a(1)	a(2)	a(3)	a(4)	a(5)	a(6)	a(7)	a(8)
4	7	9	10	16	17	21	30

b(1)	b(2)	b(3)	b(4)	b(5)	b(6)	b(7)	b(8)
5	12	13	18	20	22	24	25

c(1)	c(2)	c(3)	c(4)	c(5)	c(6)	c(7)	c(8)	c(9)	c(10)	c(11)	c(12)	c(13)	c(14)	c(15)	c(16)
4	5	7	9	10	12	13	16	17	18	20	21	22	24	25	30

A programban fogunk használni további két változót – ai és bi, melyek fogják jelölni, hogy az a ill. b tömb éppen melyik eleménél járunk.

Az ai, bi változókat a program elején beállítjuk 1-re, majd a programban (a c tömb létrehozására szolgáló ciklusban) mindig az a tömb ai-dik eleme vagy a b tömb bi-dik eleme közül a kisebbet tesszük át a c tömbbe (és növeljük az ai vagy bi értékét egyel attól függően melyik tömbből raktuk át az elemet a c tömbbe). A c tömbhöz is bevezetünk egy ci változót, mely azt fogja jelölni, hogy éppen hol járunk a c tömbben (kezdetben ennek az értéke is 1 lesz, majd minden egyes átrakásnál növeljük egyel).

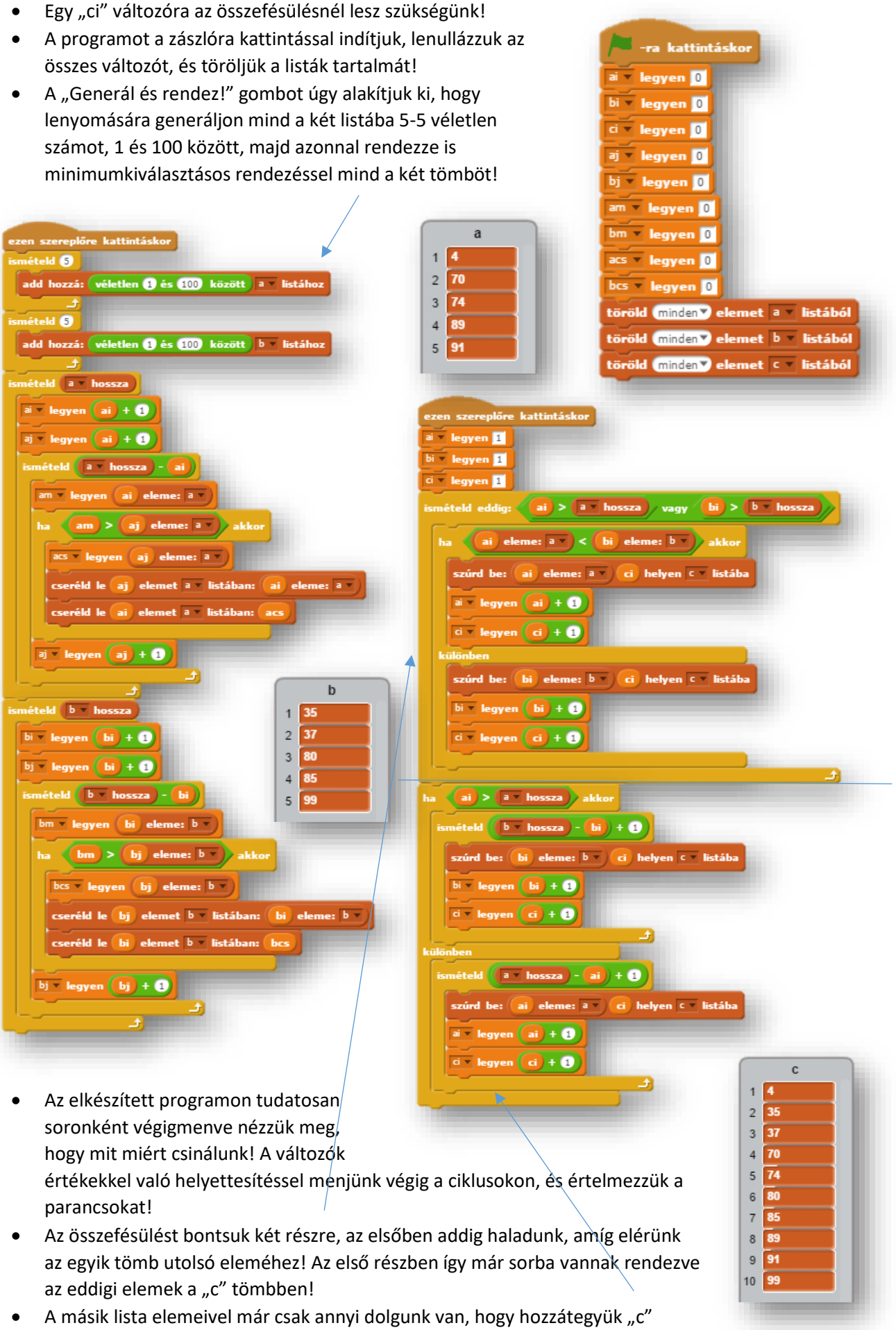
A fent leírt, elemek átrakására szolgáló algoritmust addig fogjuk ismételni, amíg nem érünk az a vagy a b tömb végére (ai vagy bi nem éri el a tömb végét, tehát amíg nem lesz több, mint az adott tömb elemeinek száma - an, bn). Ehhez egy „ismételd eddig” ciklust használunk. Ez után már csak a másik tömbből (amelyiknél még nem értünk a tömb végére) átrakjuk az összes megmaradt elemet a c tömbbe (ci-től kezdve a végéig).

32.) Gyakorlati feladat:

Nézzük meg gyakorlatban az „összefésülést”! Gombnyomásra hozzunk létre két darab 5 elemű rendezett listát! A listák neve legyen „a” és „b”! Majd egy „c” nevű listába növekvő sorrendbe átmásoljuk megfelelő egymás utáni sorrendben a listák elemeit! A feladat megoldásához nagyon sok változóra is szükségünk lesz!

- A program neve legyen: 59_splice!
- Hozzuk létre a 3 listát a megadott neveken, és készítsük el a két nyomógombot a minta szerint a képen látható felirattal!
- Az „a” lista rendezéséhez szükség van „ai”; „aj”; „am”; „acs”, változókra!
- Az „b” lista rendezéséhez szükség van „bi”; „bj”; „bm”; „bcs”, változókra!

- Egy „ci” változóra az összefésülésnél lesz szükségünk!
- A programot a zászlóra kattintással indítjuk, lenullázzuk az összes változót, és töröljük a listák tartalmát!
- A „Generál és rendez!” gombot úgy alakítjuk ki, hogy lenyomására generáljon mind a két listába 5-5 véletlen számot, 1 és 100 között, majd azonnal rendezze is minimumkiválasztásos rendezéssel mind a két tömböt!



- Az elkészített programon tudatosan soronként végigmenve nézzük meg, hogy mit miért csinálunk! A változók értékekkel való helyettesítéssel menjünk végig a ciklusokon, és értelmezzük a parancsokat!
- Az összefésülést bontsuk két részre, az elsőben addig haladunk, amíg elérünk az egyik tömb utolsó eleméhez! Az első részben így már sorba vannak rendezve az eddigi elemek a „c” tömbben!
- A másik lista elemeivel már csak annyi dolgunk van, hogy hozzátegyük „c” tömb végéhez, mert azok már eleve rendezve vannak!

27. LECKE / LISTÁKKAL VÉGZETT MŰVELETEK / ÖSSZEFOGLALÁS

28.) Logikai feladat:

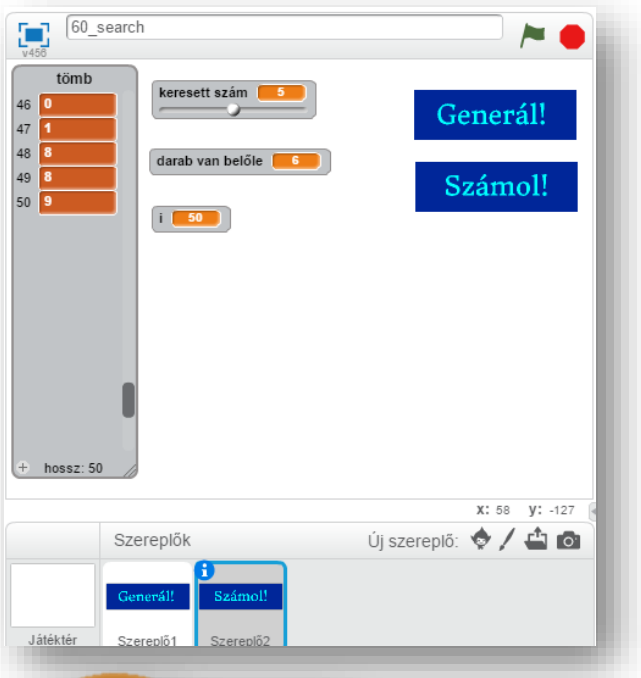
Egy születésnapi összejövetelen mindenki mindenkivel koccint. 15 koccintás történt. Hányan vannak a bulin?

Megoldás: _____ fő

33.) Gyakorlati feladat:

Állítsunk elő egy listában 50 darab 1 és 10 közötti számot! Majd egy „csúszkára” állított változóba kérjünk be egy szintén 0 és 10 közötti számot! Ezzel a számmal azt adjuk meg, hogy melyik számot számolja meg, hogy hány darab van belőle az 50 elemű listában!

- A program neve legyen: 60_search!
- A lista neve legyen: tömb; az egyik változó: „keresett szám”, a másik: „darab van belőle”; a harmadik: „i” legyen!
- A két szereplő gombot a szokott módon hozd létre a minta alapján!
- Készítsük el a leírtak alapján a programot!
- Mentsd a megadott helyre!



28.) Önálló feladat:

Generálj egy listába 30 darab 0 és 99 közötti véletlen számot! Majd csúszkára állított változóba kérjél be egy szintén 0 és 99 közötti számot! Majd keresd meg, hogy melyik helyen, és melyik szám van hozzá a legközelebb! Tehát egy változóba írja ki, hogy a tömb hányadik helyén van a megadott számhoz legközelebb eső szám!

Segítség a megoldáshoz: szükség lesz egy másik listára, ahol a „keresett szám” – „i”-edik elemet tároljuk. Az új lista legkisebb eleme lesz a legközelebb!

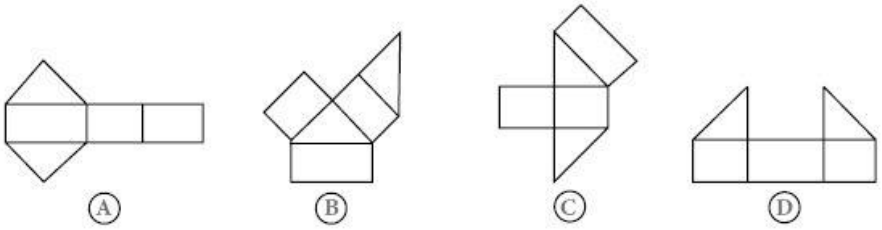
- A program neve legyen: 61_near!
- A megoldásban teljes szabadkezet kapsz!
- Mentsd a megadott helyre a kész programot!

28. LECKE / JÁTÉKOK KÉSZÍTÉSE SCRATCH PROGRAMMAL

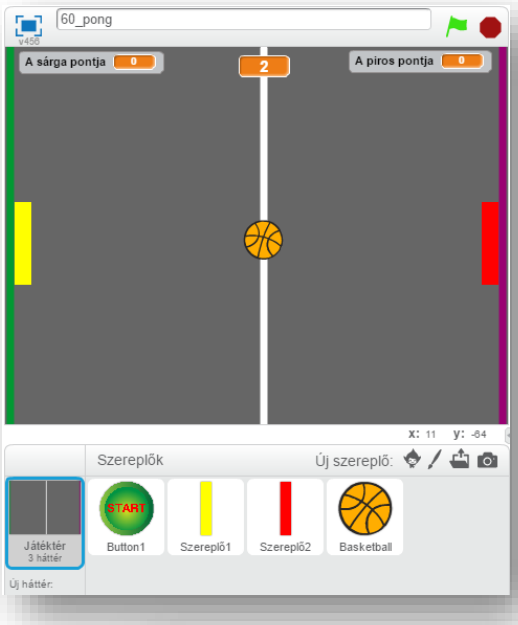
29.) Logikai feladat:

Melyik a helyes?

Megoldás: _____



34.) Gyakorlati feladat:



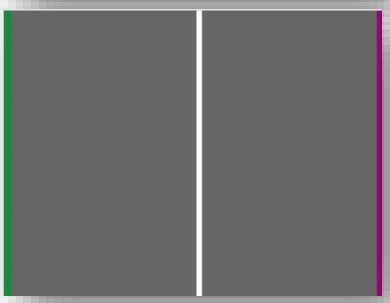
Készítsünk egy „PONG” nevű játékot! A mindenki számára ismerős játékot a következő leírás alapján készítsük el!

A játékot két játékos játssza! Egy labda véletlenszerűen pattog vissza a játéktér széléről! A két játékosnak az a feladata, hogy az ő térfelén lévő színes téglalappal elkapja a labdát! Ha nem sikerül megérinteni a szereplővel, akkor az ellenfél kap egy pontot! A játék addig megy, amíg valamelyik játékos el nem éri a 10 pontot!

A feladat megoldásához három háttérre, négy szereplőre, és négy változóra lesz szükségünk! Lépésről lépésre építsük fel a programunkat!

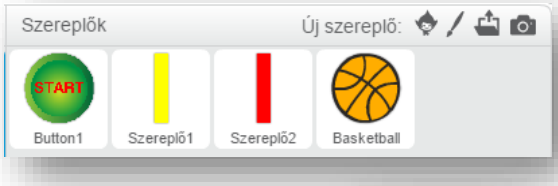
- A program neve legyen:60_pong!
- Az első háttér legyen egy kékből fehérbe átmenő felület, melyre „Marker” betűtípussal írjuk a következő szöveget a jobb oldali minta alapján! „PONG JÁTÉK - A bal oldali sárga téglalapot felfelé a „W”, lefelé az „S” billentyűkkel mozgathatod! A jobb oldali piros téglalapot a nyilakkal mozgathatod! A játék addig megy, amíg az egyik játékos el nem ér 10 pontot! Indulás előtt add meg a játék sebességét!
  A háttér neve legyen: „nyitó”!

- A második háttér lesz maga a játéktér! Egy sötétszürke háttér közepére helyezz el egy fehér közepesen vastag vonalat! A játéktér két szélére helyezzünk el két különböző színű vékony vonalat azért, hogy ha majd a labda megérintené, illetve meg tudná érinteni ezeket a színeket, akkor az ellenfél kapna egy pontot! A két szín legyen: sötétzöld és sötétlila a mint aszerint!
  A háttér neve legyen: fő!



- A harmadik háttér legyen a minta szerint ellipszis alakban színátmenetes! Bellül fehér, kívül kék! A szöveg legyen „Marker” betűtípussal: GRATULÁLOK! – A fent látható játékos nyer! (Ezt azért írjuk ki, mert majd a nyertes változója kint marad a képernyőn, a játék végén!
  A háttér neve legyen: vége!

- Mivel elég összetett, és hosszú lesz a programunk, ezért ne felejtsünk el közben menteni! Nehogy elvesszen a munkánk!



- Az első szereplő a meglévők közül legyen kiválasztva! A „Button1” szereplőre írjuk rá piros nagy betűkkel a „START” szöveget!
- A második szereplőt úgy rajzoljuk meg. Egy hosszúkás vékony citromsárga téglalap, melynek a neve legyen: sárga!
- Duplikáljuk az előző szereplőt, majd színezzük át pirosra, és nevezzük is el: piros-nak!
- A negyedik szereplő a meglévők közül legyen a „Basketball”!
- Szükségünk lesz egy olyan változóra, mellyel a sárga játékos pontjait számoljuk! A változó neve legyen: „A piros pontja”
- A piros játékos pontjait a „A sárga pontja” nevű változó számolja!
- Aztán egy „sebesség” nevű változóval tudom majd beállítani a labda gyorsaságát, mellyel nehezítem, vagy könnyítem a játékot! Ennek a változónak „csúszkát” állítunk be, amely határai 5 és 10 között lesznek!
- A „visszaszámlálás” változóra a játék indításánál lesz szükségünk! Melynek kinézetét „nagy”-ra állítjuk!

- A változókat elhelyezzük a játéktéren a következők eszerint! „A sárga pontja” változót a bal felső sarokba, „A piros pontja” változót jobb felső sarokba helyezzük! A „visszaszámlálás” változót fent középre helyezzük, a „sebesség” változót alulra középre tegyük!
- A nyitóképernyőn a „START” gombot helyezzük el, csak körülbelül lent középre, mert a későbbiekben úgyis megadjuk pontosan a helyét!
- A sárga téglalapot ballra, a pirosat jobbra tegyük,a labdát pedig középre! A pontos helyzetüket később adjuk meg!
- Kezdjünk programozni a „START” szereplő feladatainál!
- A zászlóra kattintáskor a háttér legyen „nyitó”! A szereplő jelenjen meg az (35;-120) pozícióban! A „sebesség” változó jelenjen meg, és nullázzuk le az értéket! A többi változót rejtsük el, mert most még nem használjuk őket!
- A „START” gombra kattintással kezdjük a játékot! El kell tüntetnünk a szereplőt, hátteret állítsuk „fő”-re! A „sebesség” változót tüntessük el, a „visszaszámlálót jelenítsük meg! A tényleges kezdéshez küldjön egy”start” nevű üzenetet!
- A „sárga” szereplő a zászlóra kattintáskor tűnjön el, hiszen a nyitóképernyőn még nem szerepel!
- Csak a „start” üzenet érkezésekor jelenjen meg, és azonnal ugorjon a (-219;-7) pozícióba! Illetve olyan pozícióba, hogy jobb oldalon a zöld vonal előtt középen helyezkedjen el!
- Jelenjenek meg a pontokat számláló változók!
- Készítsük le a visszaszámlálás algoritmusát, úgy, hogy 4-től számoljon vissza másodpercenként! A visszaszámlálás végén a változó tűnjön el!

- Készítsünk olyan algoritmust az előzőek alá, mellyel 10 képpontonként felfelé vagy lefelé haladunk a „W” és „S” billentyűkkel! Az egyik játékos ezzel a két billentyűvel játszik!

- Az előző alapján elkészítjük, vagy duplikáljuk, és megváltoztatjuk a „piros” algoritmust! Melyben kihagyjuk a visszaszámlálást, a helyére csak egy 5 másodperces várakoztatást szúrunk be, és a felfelé, illetve lefelé mozgást a kurzormozgató nyilakkal oldjuk meg!
- Majd a későbbiekben egy „befejezés” üzenet érkezésekor eltüntetjük a szereplőket!

- A főszereplő labda, a zászlóra kattintásnál el kell, hogy tűnjön, és a méretét lecsökkentem 50%-ra!
- A „start” üzenet érkezésekor a labda szereplő, megjelenik, és a kezdőpozícióba ugrik, középre!
- A pontszámláló változókat lenullázzuk!
- A visszaszámlálás miatt, késleltetjük 5 másodpercet a labda indulását!
- A labdát véletlen irányba fordítjuk az induláshoz!
- Folyamatosan futó algoritmust indítunk, melyben megadjuk, hogy ha a játéktér széléhez ért, akkor pattanjon vissza!
- Majd a beállított sebesség felhasználásával megadjuk, hogy milyen sebességgel lépjen a labda! („sebesség”*1)
- Ezután jöjjön a lényeg. Ha eléri a labda a két oldalon lévő színes csíkot, akkor az ellenfél változójának értéket növeljük eggyel!
- Ha pedig sikerül elérni a szereplőkkel a labdát, akkor pattanjon vissza a megfelelő véletlen nagyságú szögben a labda! A labda nézzen (45;135) és (-45;-135)° fokos irányba!
- Ha valamelyik változó értéke elérte a 10 pontot, akkor küldjön egy „befejezés” üzenetet! Ezt egy „vagy” vizsgálattal oldjuk meg!

- A „befejezés” üzenet érkezésekor (mivel a labda feladatainál vagyunk) tűnjön el a szereplő!
- A háttér legyen a „vége” nevű!
- Majd ha a piros pontja egyenlő tízzel, akkor a sárga változóját tüntessük el, különben a pirosat! Tehát a nyertes változója marad a képernyőn, így tudjuk meg, hogy ki nyert!
- Végül mindent állítsunk le!
- A programot folyamatos teszteléssel készítsük, és közben mentsünk is, nehogy elvesszen a munkánk, ha probléma adódik!

29.) Önálló feladat:

Ebben a feladatban a népszerű PAC-MAN játékot kell készíteni, a következő leírás alapján!

A játék lényege, hogy egy sárga kétállású kör alakú szereplővel haladsz egy labirintusban, és az ott elhelyezett fehér kör alakú szereplőket kell összegyűjtened, amiért pontokat kapsz! De közben, több különböző színű kis szellem is mászkál a labirintusban véletlenszerűen. (Az egyszerűség kedvéért a szellemek átmehetnek a falakon! A te szereplőd nem érhet a labirintus falához, mert akkor, vagy vissza ugrik a kezdőpontba és kezdheted előről, vagy leblokkol 2 másodpercre, viszont akkor elérheti a szellem. (Ezt döntsd el te, hogy melyik verziót készíted el!) A játékot úgy készítsd el, hogy legyen három életed! (A jobb oldali kép csak egy minta, nem kell ugyanígy kinéznie a pályának!)

- A játék neve legyen: 61_pac_man!
- A programot úgy készítsd el, hogy legyen benne kezdőképernyő, akciótér, záró képernyő!
- A szereplőket te rajzold meg, de a háttereket letöltheted az Internetről! Grafikai program segítségével változtathatsz a képeken!
- A játékban minimum 10 összegyűjtendő fehér kör legyen!
- Minimum 4 különböző színű szellem akadályozzon a gyűjtésben!
- Időkorlát nem kell, de a három étetet jelezd a képernyőn valahol!
- A szereplők, hátterek, változók elnevezésében, szabadkezet kapsz!
- A program elkészítésében mindenben a saját belátásod szerint járj el!
- A programot folyamatos teszteléssel készítsd, és közben ments, nehogy elvesszen a munkád, ha probléma adódik!

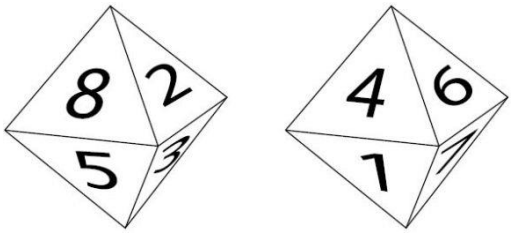
29. LECKE / SEGÉD- ÉS ALKALMAZÓI PROGRAMOK KÉSZÍTÉSE SCRATCH-BEN

30.) Logikai feladat:

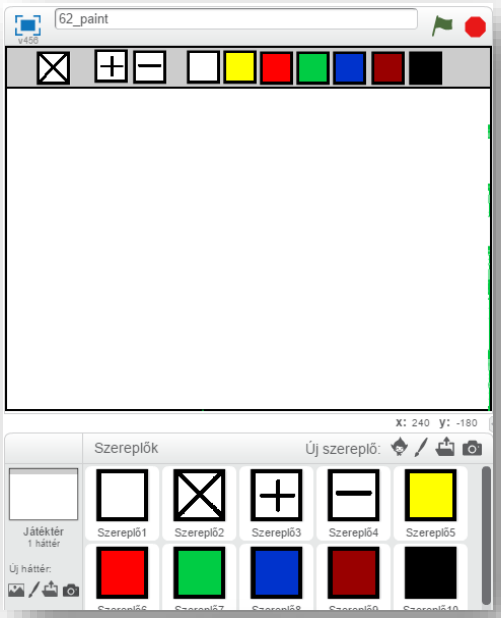
A képen egy test elejét és hátulját látod.

Melyik szám van az 5-el szemben átlósan? _____

Melyik szám van az 2-vel szemben átlósan? _____

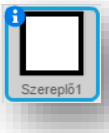


35.) Gyakorlati feladat:



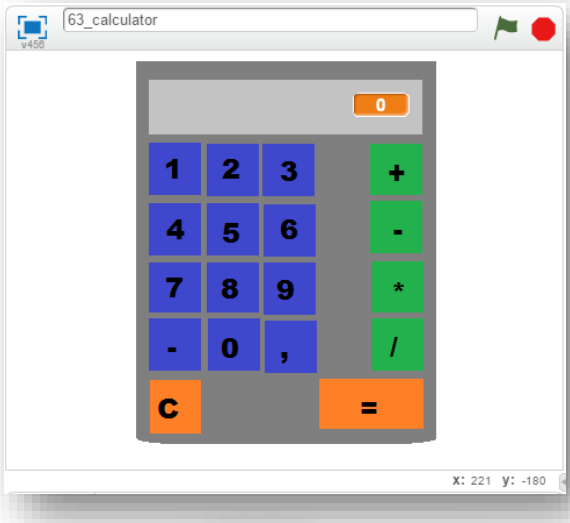
Ebben a feladatban egy egyszerű rajzoló programot fogunk készíteni! A funkciók, amelyet tudnia kell a programnak: törölni az előző rajzokat; a rajzoló „toll” vastagságát növelni, illetve csökkenteni; különböző alapszínek kiválaszthatósága!

- A program neve legyen: 62_paint!
- Egy háttérre lesz szükségünk. Rajzoljunk egy vékony fekete keretet a játéktérnek, és a felső sávra húzzunk egy fekete egyenes vonalat (kb. y:220 kp)! Színezzük be világos szürkére a felső sávot! A háttér neve legyen: alap!
- 12 szereplőt kell létrehoznunk! Először „alapgomb” néven egy fehér nézetet, fekete szegéllyel hozunk létre! Ezt fogjuk duplikálni tízszer! Mindegyiknek külön nevet adunk pl.: törlés, növel, csökkent, sárga, piros, …, fekete! Aztán megrajzoljuk a „gombok” egyéb vonalait, színeit! (Ha végeztünk a szereplőkkel, akkor beállítjuk, hogy zászlóra kattintáskor az „alapgomb” szereplő tűnjön el, hiszen erre nem lesz szükségünk!)



- A tizenkettedik szereplő, egy kis fekete kör, amivel rajzolni fogunk! A neve legyen: tollhegy! Ehhez a szereplőhöz fogjuk majd a program törzsét elkészíteni!
- De előtte az összes többi szereplőnek megadjuk, hogy ha rákattintunk a „gombra”, mekkora legyen a mérete %-ban, hogy elférjen a felső sávon! Majd felhelyezzük a helyére!
- Ebben a programrészletben küldjön egy gombra utaló üzenetet! Például: törölj, növelj, csökkents, fehér, sárga, … , fekete! Ezt tízszer megtesszük!
- A tollhegy szereplőnél a zászlóra kattintással először töröljük az eddigi rajzokat, és a toll méretét állítsuk 1 –re!
- Az összes üzenet érkezésekor beállítjuk a teendőket! „Törölj” üzenet érkezésekor, letisztítja a rajfelületet! A „növeld” üzenet érkezésekor a tollméret változzon 2 kp-al, stb.!
- Majd a program fő részénél „mindig” algoritmus futása közben vizsgálja, ha az egér le van nyomva, ugorjon az egérmutató helyére és tegye le a tollat! Ha felengedjük az egeret, akkor emelje fel a tollat! Csak akkor tudjunk rajzolni, ha a felső sáv alatt van az egér!
- Programozás közben teszteljünk és mentsük!

30.) Önálló feladat:



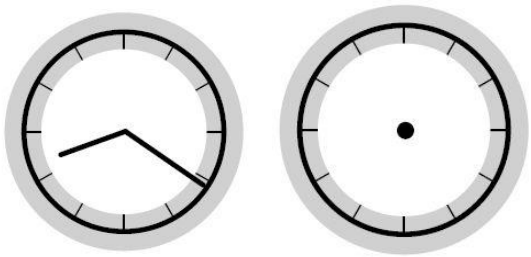
Ebben a feladatban egy egyszerű „számológép”-et kell készíteni a következő leírás alapján! Csak két számmal végzett műveletet kell megoldani a programmal!

- A program neve legyen 63_calculator!
- Először a hátteret kell elkészítened, amely egy egyszerű sötétszürke téglalap, melyen egy világosszürke kisebb téglalap van a tetején a minta szerint!
- 18 darab szereplőt kell létrehoznod! (1-9, a 0 számokat; a műveleti jeleket -,+,*,/; az egyenlőség jelet; és a „C” törlő gombot)
- Szükség lesz legalább 4 változóra! („1_szám”; „2_szám”; „művelet”; és az „eredmény” változókra)
- A számokat úgy kell bevinni, hogy ha rákattintunk egy szereplőre (pl.: 2), akkor küldjön üzenetet!
- Ha érkezik az előbbi üzenet, akkor vizsgálja, hogy melyik számnál vagyunk, és az „együtt __ __” parancsot használjuk az „összefűzéshez”! Ezt minden gombra el kell készítenünk! (Ezeket elkészítheted a háttér feladatainál például!)

- A „C” gombra kattintásnál minden változót töröljön, illetve nullázzon le!
- Az egyenlőség jelre kattintással hozza működésbe a törzsprogramot, melyben „ha / akkor / különben” parancsokkal vizsgálja a műveleti jeleket, és számolja ki a megoldásokat!
- A hibajelzéseket is kezelve, ha túl nagy számot, számokat nem tud kezelni, és hibajelzést küld a program (Infinity), akkor írja ki, hogy: „Próbáld újra!”
- Minden kérdéses egyéb esetben szabadkezet kapsz!
- A programot folyamatos teszteléssel készítsd, és közben ments, nehogy elvesszen a munkád, ha probléma adódik!

30. LECKE / ZÁRÓPROJEKT FELADAT

31.) Logikai feladat:



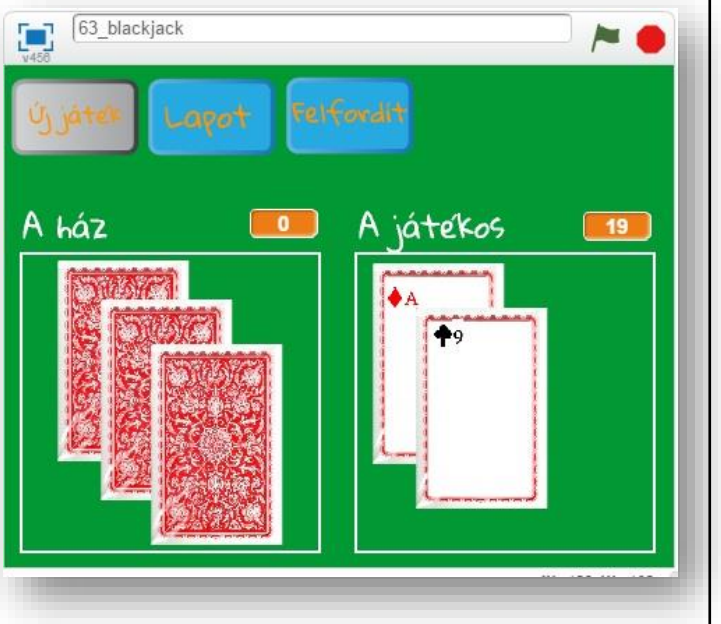
Zedország egyik látványossága a Tükörmúzeum.

A múzeum különlegessége, hogy minden tárgyat úgy látunk, mintha tükörben néznénk azokat. A következő képen található órát a múzeum egyik termében lehet megtekinteni. Tulajdonképpen, akkor most mennyi az idő?

Megoldás: _____

A mellette lévő órán pedig rajzold be, hogy mit fog mutatni az óra 4 óra 15 perc múlva!

31.) Önálló feladat:



Ebben a feladatban „BlackJack” kártyajátékot kell készíteni.

(A jobb oldali kép csak egy minta, nem pont így kell kinéznie a programnak.)

A kártyapakliban a számozott lapok értéke megegyezik az adott számmal, amelyen figura található, azaz, a bubik, a dámák és a királyok értéke egyenként 10 pont. Az ‘Ász’ az egyetlen olyan lap, amelynek kétfajta értéke van, így az ász érhet 11 vagy 1 pontot is, attól függően, hogy milyen kártyakombinációja van a játékosnak. Például, ha a játékosnak osztottak egy ászt és egy 9-est, akkor valószínűleg 11-nek számolja az ászt, így a kártyái értéke 20 pont (11+9) lesz. Viszont, ha a játékos egy 3-t és egy ászt kap, lehet, hogy az ászt 1-nek veszi és kér még egy lapot az osztótól.

A figurás lapok és az ász értéke: A figurás lapok 10 pontot, míg az Ász 1 vagy 11 pontot ér. Egy figurás lap és egy Ász kombinációja tehát 21-et ad.

A blackjackban, a játék lényege, hogy a 21-hez lehető legközelebbi pontszámot érjünk el. Ezt úgy kell elérni, hogy ne haladjuk meg ezt a számot, különben veszítünk. Ha nem érjük el a 21 pontot, a cél az, hogy magasabb értéket érjünk el, mint az osztó. Például, ha egy játékosnak egy 10-est és egy 8-ast osztanak, a lapjai összértéke 18. Az osztónak van egy 9-es és egy 8-as lapja, melyek összértéke 17. Ebben az esetben, a játékos pontszáma magasabb, mint az osztóé és közelebb van a 21-hez, így a játékos nyer.

Leegyszerűsítve, a legfőbb cél a blackjack játékban egy Ász és egy 10-et érő lap szerzése, és a 21 pont elérése.

Osztás: Az osztó a játékosnak oszt egy lapot, a színével felfelé, majd egyet saját magának is, szintén a színével felfelé. Ezután ismét oszt a játékosnak egy újabb lapot a színével felfelé, valamint saját magának is, de ezt már színével lefelé. Ezt a lapot a játékos nem láthatja.

Lapkérés: A játékos az osztást követően lapot kérhet (Hit), megállhat (Stand), illetve speciális döntéseket hozhat (Double, Split, Surrender). A játékosnak be kell fejeznie a játékot, az osztó csak ezután kérhet magának lapot. Több játékos esetén mindig csak az egyik játékos játszik, és ha befejezte a játékot, akkor a következő játékos következik.

Ha a játékos megállt, valamint több játékos esetén az utolsó játékos is befejezte a játékot, akkor kezdhet az osztó saját magának lapot osztani. Az osztónak 16-nál kötelezően lapot kell kérnie, és 17-nél már kötelezően meg kell állnia. Az osztó az Ász értékét nem tekintheti 1-nek, ha a lapok összértéke az Ász 11-es értékével számolva 17 és 21 közötti. Csak akkor tekintheti 1-nek, ha a lapok összértéke az Ász 11-es értékével számolva meghaladná a 21-et.

A játék végeredménye:

Ha a játékos lapjainak összértéke közelebb van a 21-hez, mint az osztóé, akkor a játékos a tétet 2:1 arányban kapja meg.

Ha az osztó lapjainak összértéke közelebb van a 21-hez, mint a játékosé, akkor a játékos elvesztette a tétet.

Ha a játékos és az osztó lapjainak összértéke egyforma, akkor az állás döntetlen (Push), a megtett tétet visszakapja a játékos.

Ha a játékos lapjainak összértéke a játék során a 21-et meghaladja (Bust), akkor a játékos elvesztette a tétet, az osztó későbbi eredményétől függetlenül.

Ha az osztó lapjainak összértéke a játék során a 21-et meghaladja (Bust), akkor a játékos a tétet 2:1 arányban kapja meg.

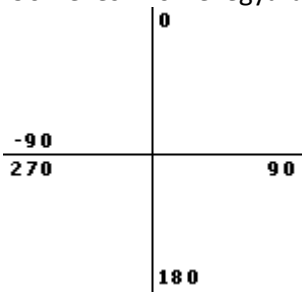
Ha a játékos az első két lapjának összértéke pontosan 21 (Blackjack), és az osztó nem Blackjack-et ért el, akkor a játékos a megtett tétet 3:2 arányban kapja meg.

A fent leírtak alapján kell elkészítened a programot.

- A program neve legyen: 63_blackjack!
- A teljes program elkészítésében szabad kezet kapsz!
- Programozás közben tesztelj és ments!

PARANCSLISTA

Mozgás

	Elmozdítja a szereplőt előre vagy hátra. **Megjegyzés:** Ez a fajta mozgás pillanatszerű: olyan, mintha a szereplő nagyon gyors volna, és csak az indulás, és az érkezés pillanatában készülne róla fénykép. Ha azt szeretnéd, hogy a mozgás folyamatos legyen, használj ismétlést:  Másik lehetőség: használd a **csússz** parancsot.
	Elforgatja a szereplőt az órajárással egyező irányban.
	Elforgatja a szereplőt az órajárással ellentétes irányban.
	A megadott irányba fordítja a szereplőt (0=fel, 90=jobbra, 180=le, -90=balra; de ezektől különböző számok is megadhatók). **Megjegyzés:** Az irányokat *a függőleges egyeneshez* viszonyítva lehet megadni. A fokszámok 0-tól az óramutató járása szerint folyamatosan nőnek, azzal ellentétes irányban haladva pedig csökkennek. Így például a nyugati irány -90-nel és 270-nel egyaránt kifejezhető. 
	Az egérmutató, vagy egy másik szereplő felé fordítja a szereplőt.
	A játéktér megadott x-y helyére helyezi át a szereplőt.
	Az egérmutató, vagy egy másik szereplő helyére helyezi át a szereplőt.
	Eltolja a szereplőt a játéktér megadott x-y helyére a megadott idő alatt. **Megjegyzés:** Ezt a parancsot könnyen használhatod úgy is, hogy a megérkezés helye függjön a szereplő jelenlegi helyétől. Például ez mindig egy átlós irányú csúszás: 

x változzon: 10	Megváltoztatja a szereplő x helyét a megadott értékkel.
x legyen 0	Beállítja a szereplő x helyét a megadott értékre.
y változzon: 10	Megváltoztatja a szereplő y helyét a megadott értékkel.
y legyen 0	Beállítja a szereplő y helyét a megadott értékre.
ha szélen vagy, pattanj vissz	Elforgatja a szereplőt az ellenkező irányba, amikor megérinti a játéktér szélét. **Megjegyzés:** Az ellenőrzés, tehát hogy a szereplő a játéktér szélén van-e, csak a parancs végrehajtásának pillanatában történik meg. Ezért szinte mindig ismétléssel célszerű használni: 
x hely	Megadja a szereplő x helyét (-240-től 240-ig terjedő érték).
y hely	Megadja a szereplő y helyét (-180-tól 180-ig terjedő érték).
irány	Megadja a szereplő irányát (0=fel, 90=jobbra, 180=le, -90=balra).

Kinézet

jelmez legyen jelmez2	Megváltoztatja a szereplő kinézetét úgy, hogy egy másik jelmezt állít be. **Megjegyzés:** A jelmezekre a sorszámukkal is lehet hivatkozni, így különböző változó értékeket húzhatsz a jelmez-megnevezés helyébe: 
válts jelmezt	A jelmezlista következő jelmezét állítja be (ha elérte a jelmezlista végét, visszatér az első jelmezhez).
jelmez	Megadja a szereplő által viselt jelmez sorszámát.
mondd: Szia! 2 mp-ig	Megjeleníti a szereplő szövegbuborékját a megadott ideig. **Megjegyzés:** Szövegen túl változó értékeket is megjeleníthetsz így.  Persze nem ez a változó értékek megjelenítésének legegyszerűbb módja (minden változó érték megjeleníthető a neve melletti szürke négyzetre kattintással).

mondd: Szia!	Megjeleníti a szereplő szövegbuborékját (eltüntetheted a szövegbuborékot, ha ezt a parancsot üresen adod ki).
gondold: Hmm... 2 mp-ig	Megjeleníti a szereplő gondolatfelhőjét a megadott ideig.
gondold: Hmm...	Megjeleníti a szereplő gondolatfelhőjét.
szín hatás változzon 25	Megváltoztatja a kiválasztott grafikus hatást a megadott értékkel (válassz egy hatást a legördülő menüből).
szín hatás legyen 0	Beállítja a kiválasztott grafikus hatást a megadott értékre (a legtöbb hatás lehetséges értékei 0-tól 100-ig terjednek).
töröld a hatásokat	A szereplő minden grafikus hatását törli.
méret változzon: 10	Megváltoztatja a szereplő méretét a megadott értékkel.
méret legyen 100 %	Beállítja a szereplő méretét a megadott érték szerint (amely az eredeti mérethez viszonyított százalékos arány).
méret	Megadja a szereplő méretét (az eredeti méret százalékában).
jelenj meg	Láthatóvá teszi a szereplőt a játéktérben.
tűnj el	Láthatatlanná teszi a szereplőt **Megjegyzés:** Ha egy szereplő láthatatlan, a többi szereplő nem képes őt érzékelni az *érinted?* elemmel, és ő sem képes (akár látható) társait érzékelni.
kerülj legelőre	A szereplőt az összes többi elé hozza.
kerülj 1 szinttel hátrébb	Hátraküldi a szereplőt a megadott számú szinttel, így más szereplők kitakarhatják.

Hang

játszd le: miaú	Elindítja a legördülő menüből kiválasztott hang lejátszását, és azonnal a következő parancsra ugrik (akkor is, ha a hang még szól).
játszd le: miaú és várd meg	Lejátszik egy hangot, és csak akkor ugrik a következő parancsra, ha a hang lejátszása befejeződött.
minden hangot állíts le	Leállítja az összes most hallható hangot (ezután megszólaltathatók újabbak).
dobolj: 48 0.2 ütemig	Megszólaltatja a legördülő menüből kiválasztott dobot a megadott ütemig.

szünetelj 0.2 ütemig	Várakozik a megadott ütemig.
szóljon 60 0.5 ütemig	Zenei hangot szólaltat meg a megadott ütemig (a nagyobb szám magasabb hangnak felel meg).
hangszer legyen 1	Beállítja a szereplő hangszerét, amelyet a zenei hang megszólaltatására használ (minden szereplő saját hangszert használ).
hangerő változzon: -10	Megváltoztatja a szereplő hangerejét a megadott értékkel (a hangokra és a hangszerekre egyaránt vonatkozik).
hangerő legyen 100 %	Beállítja a szereplő hangerejét a megadott értékre (a hangokra és a hangszerekre egyaránt vonatkozik).
hangerő	Megadja a szereplő hangerejét (minden szereplő saját hangerőt használ).
tempó változzon: 20	Megváltoztatja a közös tempót a megadott értékkel.
tempó legyen 60 ütem/perc	Beállítja a közös tempót a megadott értékre (minden szereplő ugyanazt a tempót használja). Meghatározhatod, hogy egy percbe hány egyenlő ütem tartozzon, az ütemre hivatkozó parancsokban pedig azt, hogy ehhez képest hány ütem hosszan szólaljon meg az adott hangszer.
tempó	Megadja a közös tempót.

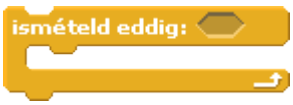
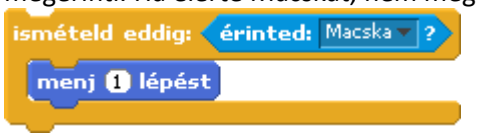
Toll

töröld a rajzokat	Az összes tollrajzot és lenyomatot törli a játéktérről.
tollat tedd le	Leteszi a szereplő tollát, így a szereplő rajzolni fog, miközben mozog.
tollat emeld fel	Felemeli a szereplő tollát, így a szereplő nem fog rajzolni, miközben mozog.
tollszín legyen	Beállítja a toll színét a palettáról kiválasztott színre.
tollszín változzon: 10	Megváltoztatja a toll színét a megadott értékkel.
tollszín legyen 0	Beállítja a toll színét a megadott értékre (0=a szivárvány vörös határa, 100=a szivárvány kék határa).
tollárnyalat változzon: 10	Megváltoztatja a toll árnyalatát a megadott értékkel.
tollárnyalat legyen 50	Beállítja a toll árnyalatát a megadott értékre (0=nagyon sötét, 100=nagyon világos).
tollméret változzon: 1	Megváltoztatja a toll vastagságát.

	Beállítja a toll vastagságát.
	Lenyomatot készít a szereplőről a játéktérre.

Vezérlés

	Lefuttatja a hozzá kapcsolt feladatot, ha a zöld zászlóra kattintanak (egy szereplőhöz több ugyanolyan eseményjelző sapka is tartozhat).
	Lefuttatja a hozzá kapcsolt feladatot, ha a megadott billentyűt megnyomják.
	Lefuttatja a hozzá kapcsolt feladatot, ha a szereplőre kattintanak.
	A megadott ideig vár, majd folytatja a következő paranccsal.
	Lefuttatja a benne elhelyezett parancsokat, majd ezt állandóan megismétli.
	Lefuttatja a benne elhelyezett parancsokat a megadott alkalommal.
	Küld egy üzenetet az összes szereplőnek, hogy valamilyen feladatot hajtsanak végre, majd a következő parancsra ugrik (anélkül, hogy a kitűzött feladatok befejezését megvárná). **Megjegyzés:** A következő példában Macska egy parancs kiadásával bármikor megkérheti Kutyát, hogy nézzen az ő irányába: Macska   Kutya  
	Küld egy üzenetet az összes szereplőnek, hogy valamilyen feladatot hajtsanak végre, és meg is várja, hogy azokat mindenki befejezze.
	Lefuttatja a hozzá kapcsolt feladatot, ha a megadott üzenet megérkezik.
	Folyamatosan ellenőrzi, hogy a megadott feltétel igaz-e, és ha igaz, lefuttatja a benne elhelyezett parancsokat. **Megjegyzés:** Vigyázz arra, hogy mindaddig amíg a feltétel igaz, az itt elhelyezett parancsokat állandó ismétléssel végre fogja hajtani a szereplő. Ha azt

	szeretnéd, hogy egy feltétel teljesülése esetén csak egy alkalommal történjen valami, akkor használd a **várj eddig** parancsot (lásd ott).
	Ha a feltétel (pillanatnyilag) igaz, lefuttatja a benne elhelyezett parancsokat.
	Ha a feltétel (pillanatnyilag) igaz, a *ha* részben elhelyezett, ha pedig nem igaz, a *különben* részben elhelyezett parancsokat futtatja le.
	A megadott feltétel teljesülésére vár, majd folytatja az utána kapcsolt parancsokkal. **Megjegyzés:** Jó szolgálatot tesz például akkor, ha egy feltétel teljesülésekor csak egyszer szeretnéd, hogy valami történjen: 
	Ellenőrzi, hogy a feltétel teljesül-e, és ha egyelőre nem teljesül, lefuttatja a benne elhelyezett parancsokat, majd újabb ellenőrzést végez. Ha már a feltétel igaz, folytatja az utána kapcsolt parancsokkal. **Megjegyzés:** A következő példában a Kutya addig menetel, amíg végül Macskát megérinti. Ha elérte Macskát, nem megy már tovább. 
	Megszakítja a feladat végrehajtását.
	Minden szereplő minden feladatát leállítja.

Érzékelés

	Igaz értéket ad, ha a szereplő érint egy megadott szereplőt, az egérmutatót, vagy a játéktér szélét (a legördülő menüből választhatsz).
	Igaz értéket ad, ha a szereplő érinti a megadott színt a játéktérben (kattints a színmintán, majd válassz egy színt a palettáról vagy a játéktérből).

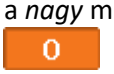
érint szín?	Igaz értéket ad, ha az első szín (a szereplőn belül) érinti a második színt (a háttérben vagy egy másik szereplőn) (kattints a színmintán, majd válassz egy színt palettáról vagy a játéktérből).
kérdezd meg: Hogy hívnak? és várj	Kérdést tesz fel, amelyre a billentyűzetről lehet válaszolni. Várakozik az Enter lenyomásáig vagy a jobb oldali pipára kattintásig. A szöveg a **válasz** elembe kerül.
válasz	Megadja a **kérdezd** parancs legutóbbi használatakor kapott választ (lényegtelen, hogy melyik szereplő tette fel a kérdést).
egér x	Megadja az egérmutató x helyét.
egér y	Megadja az egérmutató y helyét.
egér lenyomva?	Igaz értéket ad, ha az egér gombja le van nyomva.
szóköz lenyomva?	Igaz értéket ad, ha a megadott billentyű le van nyomva.
távolság:	Megadja az egérmutató, vagy egy másik szereplő távolságát a szereplőtől.
nullázd az órát	Beállítja az időmérőt nullára.
óra	Megadja az óra pillanatnyi értékét (másodpercben) (az óra mindig jár!). **Megjegyzés:** Sose vizsgáld azt, hogy az óra felvesz-e valamilyen értéket, helyette azt nézd, hogy azt már átlépte-e. Tudniillik a programodnak egyszerre sok feladatot kell végeznie, és megeshet, hogy abban a pillanatban, amikor az óra például egy percnél jár, éppen nem azzal a vizsgálattal fog foglalkozni, amelyben az óra=60 feltételt ellenőriznéd. Ezért így néz ki egy jól működő vizsgálat:  Az óra neve melletti szürke négyzetre kattintással a kijelzőt a játéktérben megjelenítheted, vagy eltüntetheted.
Szereplő1 x hely értéke	Megadja a bal oldalon kiválasztott szereplő jobb oldalon kiválasztott értékét (ezek között lehetnek saját változók is).
hangerő	Megadja a számítógéphez csatlakoztatott mikrofon által észlelt hangerőt (1-től 100-ig).

	Igaz értéket ad, ha a mikrofonba 30-nál hangosabb hang érkezik (az 1-től 100-ig terjedő tartomány szerint).
	Megadja a megadott érzékelő értékét (hogy használhasd ezt az elemet, szükséged van egy "ScratchBoard"-ra).
	Igaz értéket ad, ha a megadott érzékelő le van nyomva (hogy használhasd ezt az elemet, szükséged van egy "ScratchBoard"-ra).

Műveletek

	Összead két számot.
	Kivonja az első számból a másodikat.
	Összeszoroz két számot.
	Elosztja az első számot a másodikkal.
	Véletlenszerűen választ egy egész számot a megadott tartományból (vagy nem egészt, ha a tartomány valamelyik határa nem egész).
	Igaz értéket ad, ha az első szám kisebb, mint a második (vagy az első szöveg betűrendben előrébb van, mint a második).
	Igaz értéket ad, ha a két szám egyenlő (vagy a két szöveg megegyezik).
	Igaz értéket ad, ha az első szám nagyobb, mint a második (vagy az első szöveg betűrendben hátrébb van, mint a második).
	Igaz értéket ad, ha mindkét feltétel igaz.
	Igaz értéket ad, ha valamelyik feltétel igaz.
	Igaz értéket ad, ha a feltétel hamis, és hamisat, ha igaz.
	Egymás után ír két szöveget.
	Megadja a szöveg megadott sorszámú betűjét.
	Megadja a szöveg betűinek számát.
	Megadja az első szám másodikkal való osztásánál keletkező maradékot.
	Megadja a számhoz legközelebb eső egész számot.
	Matematikai függvények értékeit számolja ki.

Változók

	Segítségével új változót hozhatsz létre, és nevezhetsz el. Eldöntheted, hogy a változó *mindenkié* legyen vagy csak a *kiválasztott szereplőhöz* tartozzon.
	Törli a kiválasztott változót.
	Megadja a változó jelenlegi értékét. **Megjegyzés:** A változó neve melletti szürke négyzetre kattintással a változó kijelzőjét eltüntetheted, vagy megjelenítheted a játéktérben.  A megjelenő kijelzőt bárhová mozgathatod, és a jobb kattintással előhívható helyi menüben a *normál* és a *nagy* megjelenítés között is választhatsz.  Választhatod továbbá (szintén ebben a helyi menüben) a *csúszka* megjelenítése opciót, amivel lehetőséget adhatsz a programot használónak arra, hogy a változó értékét saját maga változtathassa meg a program futása alatt. Egy újabb jobb kattintással beállíthatod a csúszka lehetséges legkisebb és legnagyobb értékét is a *csúszka határainak megadásával.* 
	Beállítja a változó értékét a megadott értékre.
	Megváltoztatja a változó értékét a megadott értékkel (ha az nagyobb nullánál, akkor növel, ha kisebb mint nulla, akkor csökkent).
	Megjeleníti a változó értékét a Játéktérben.
	Eltünteti a változó értékét a Játéktérből.
	Segítségével új listát hozhatsz létre, és nevezhetsz el. Eldöntheted, hogy a lista *mindenkié* legyen vagy csak *a kiválasztott szereplőhöz* tartozzon.
	Törli a kiválasztott listát.
	Megadja a lista összes elemét.
	Felveszi a lista végére a megadott értéket. Az érték lehet szám vagy szöveg.
	Törli a lista valamelyik, vagy az összes elemét (használd a legördülő menüt, vagy írj be egy sorszámot). A legördülő

	menü *utolsó* pontját választva a lista utolsó elemét törli. A *minden* pontot választva a lista összes elemét törli. **Megjegyzés:** A törlés csökkenti a lista hosszát.
	Beszúrja a lista megadott helyére a megadott értéket (használd a legördülő menüt, vagy írj be egy sorszámot). A legördülő menü *utolsó* pontját választva a lista végére veszi fel az értéket. Az *egyik* pontot választva a lista egy véletlenszerűen választott helyére szúrja be az értéket. **Megjegyzés:** Elem beszúrásakor a korábbi elemek megmaradnak, és a lista hossza eggyel nő.
	Lecseréli a listában a megadott helyű elemet a megadott értékre (használd a legördülő menüt, vagy írj be egy sorszámot). A legördülő menü *utolsó* pontját választva a lista utolsó elemét cseréli le. Az *egyik* pontot választva a lista egy véletlenszerűen választott elemét cseréli le. **Megjegyzés:** Elem lecserélésekor a lista hossza nem változik meg.
	Megadja a lista megadott sorszámú elemét (használd a legördülő menüt, vagy írj be egy sorszámot). A legördülő menü *egyik* pontját választva a lista egy véletlenszerűen választott elemét adja meg.
	Megadja a lista elemeinek számát.
	Igaz értéket ad, ha a listában a megadott elem megtalálható (csak pontos egyezés esetén ad igaz értéket).

(A parancslista forrása: http://scratch.inf.elte.hu/lecke/kisokos/parancslista)