



Caiet de Activitati Nr 1



SCRATCH

Curs. Exerctii, Solutii, Teste si
Proiecte

Caiet de Activitati Nr 1

Introducere

Caiet de Activitati Nr 1 - Scratch - Nivel Incepator

Introducere

Scopul acestui document este de a furniza cunoștințe de programare de bază în Scratch copiilor între 7-14+ ANI fără sau cu foarte puțină experiență în codificare.

Obiectivul este ca ei să obțină definițiile și practică de bază necesară codării computerizate, independent de orice limbaj de programare.

Copii vor afla ce sunt declarațiile, variabilele, proceduri, funcții, bucle și cum pot fi manipulate. De asemenea, vor avea o primă privire la ceea ce este codarea computerizată orientată pe obiecte. În acest document, vom folosi Scratch 3.0 pentru a dezvolta algoritmi.

SCRATCH are avantajul de a permite dezvoltatorilor să implementeze algoritmi folosind instrumente de programare grafică. În fiecare secțiune sunt propuse exerciții folosind Scratch. Așoritate exercițiilor sunt ușoare și permit înțelegerea conceptelor.

Cu toate acestea, au fost inserate și exerciții notate cu stele ** mai complicate pentru sfarsit. Scratch este un software și un portal web gratuit dezvoltat de MIT pentru a le permite copiilor să învețe cum să dezvolte povești interactive și animații într-o manieră colaborativă.

Într-un proiect Scratch, poți controla spritele și le poți face să se miște, să interacționeze ...Limba propusă pentru controlul acestor sprite, conține toate structurile de bază necesare în orice algoritmi.

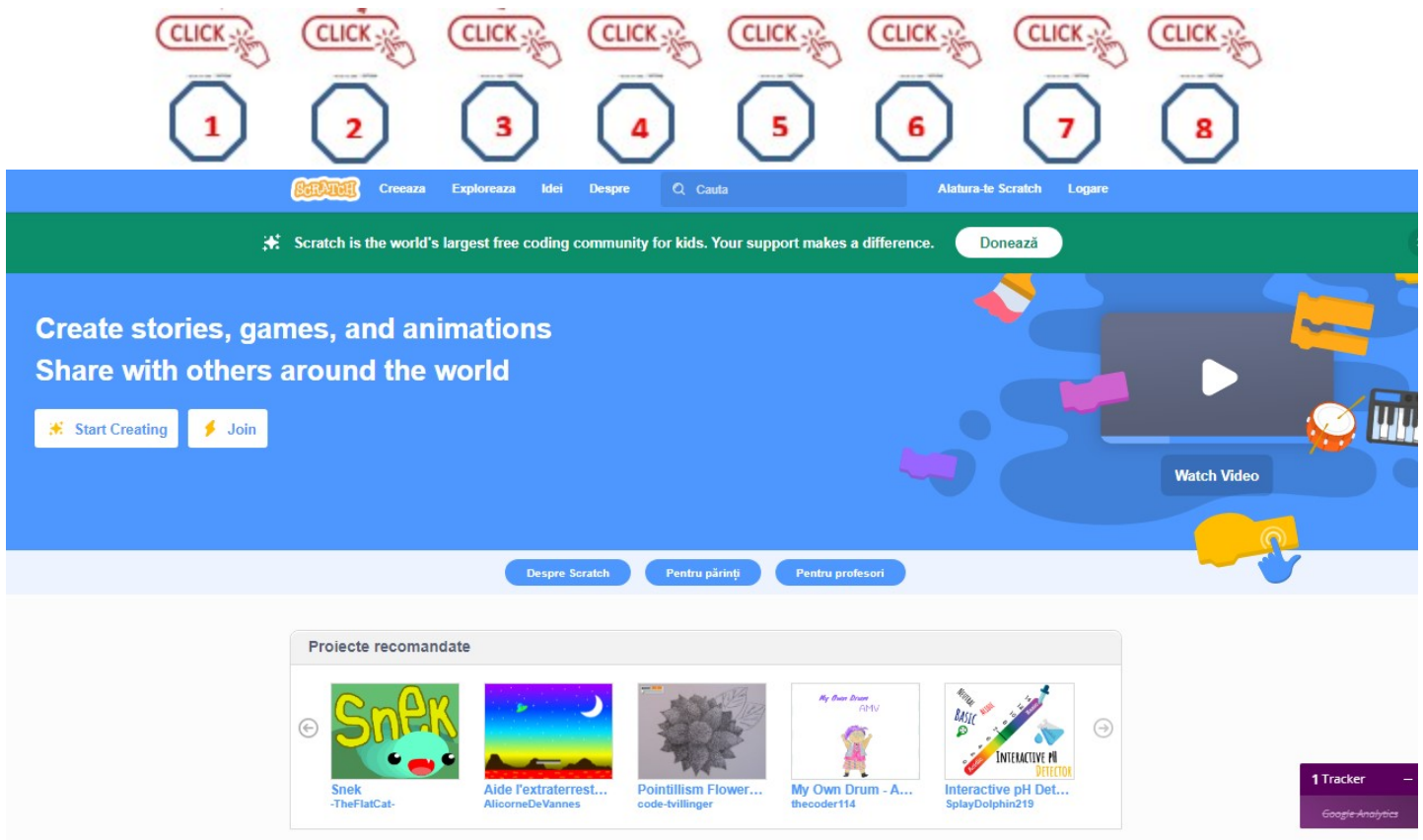
Scratch poate fi folosit pentru a dezvolta orice fel de algoritmi. În continuare, pentru fiecare parte vor exista exerciții care vizează controlul spriteelor, dar exerciții, de asemenea, pentru a proiecta algoritmi clasici mai teoretici.

Instrumentul este disponibil online la adresa: <https://www.scratch.mit.edu>

În plus, este disponibil un editor offline: <https://scratch.mit.edu/scratch2download>

Când accesați site-ul web <https://www.scratch.mit.edu> puteți crea sau nu, un cont gratuit pentru a vă salva proiectele. Poti crea un proiect nou făcând clic pe meniul dedicat (Creare), chiar dacă nu ai un cont.

Introducere



1



Menu Genaral Scratch

2



Creaza Proiect Nou Scratch

3



Expolreaza Proiectele Scratch incepand cu cel mentionat in Cauta

4



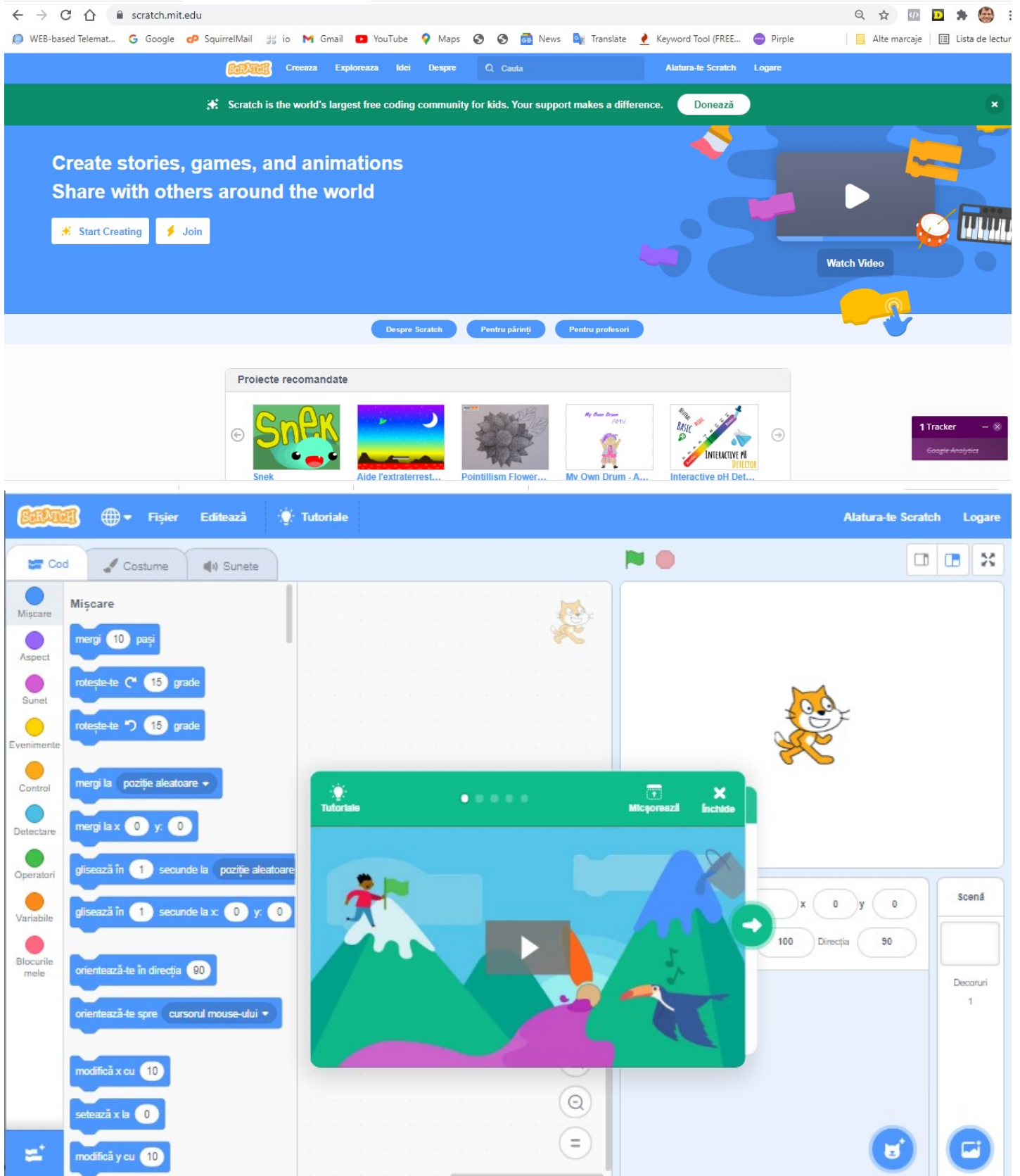
Inscriete in Scratch



Conectare Cont Scratch

Introducere

Introducere

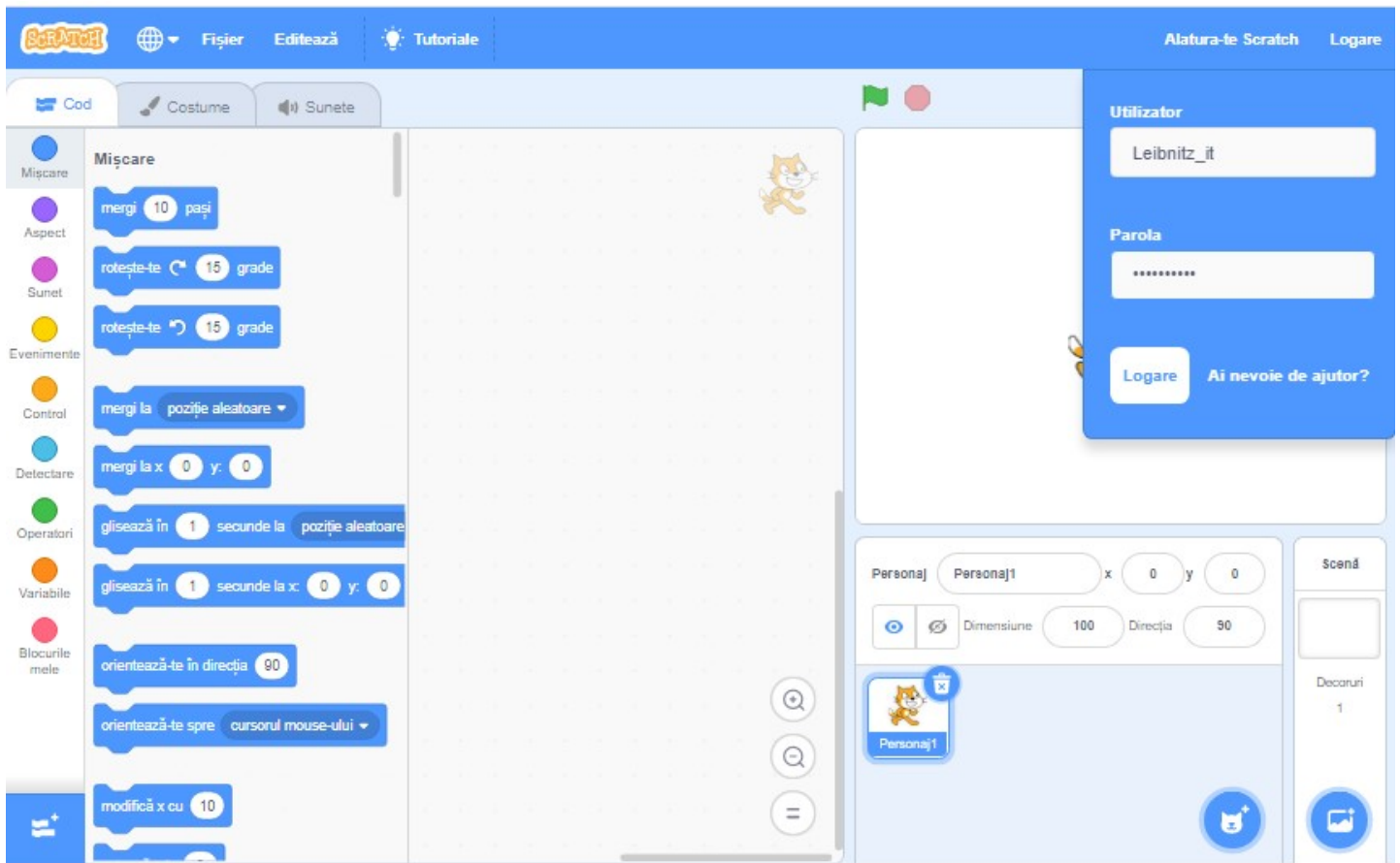


Avantajul creerii unui cont este posibilitatea de a salva toate proiectele si activitatile din momentul creeri contului si reconectarea la proiectele salvate, dupa semnare

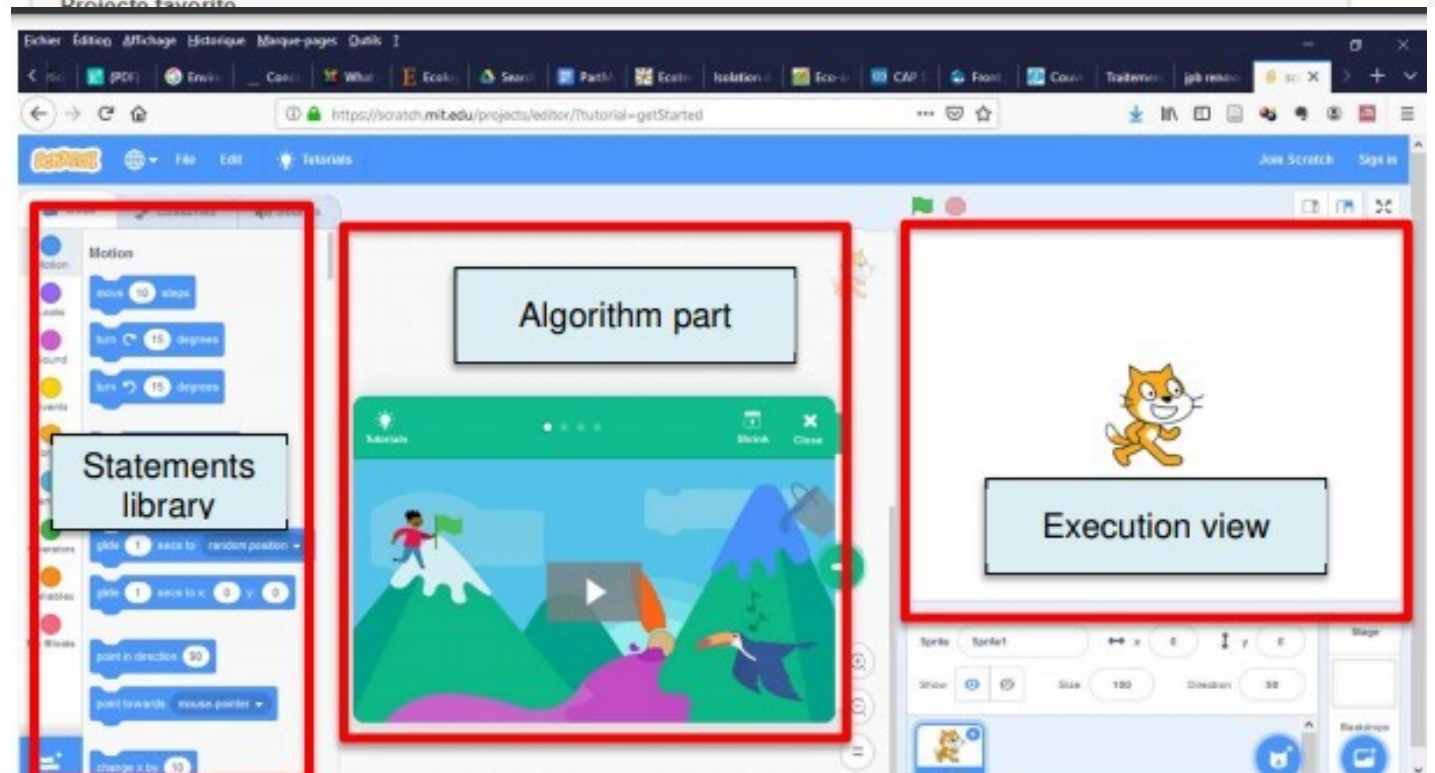
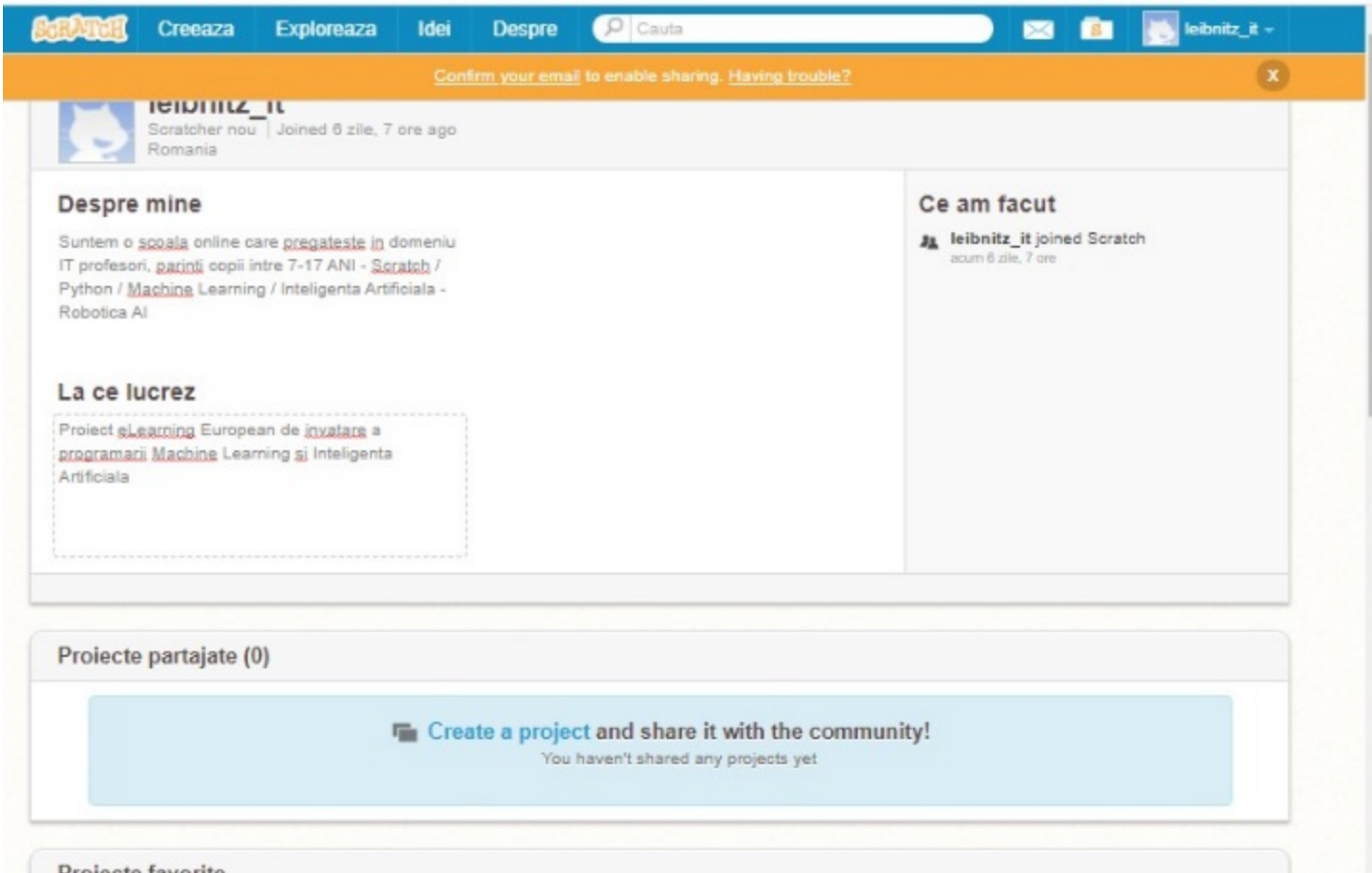
Introducere

Introducere

Odata creat un cont – poti vedea profilul contului ca mai jos:



Introducere



Introducere

Introducere

După ce ați creat un nou proiect, accesați interfața de unde vă puteți crea algoritmi și observa rezultatul execuției lor (cu sprite-ul) e instantanee în partea Scena din dreapta (Execution view).

Cele trei părți principale ale interfeței sunt:

1. **Biblioteca de Instrucțiuni** (partea stanga)
2. **Instrucțiuni** (lista de instruction pe care programatorul le dă calculatorului)
3. **Scena** (spatiu de executie al instructiunilor)

Pentru a construi un algoritm, trebuie doar să glisăm și să fixăm elemente din biblioteca de blocuri în partea algoritmului (în centru). Biblioteca conține un set imens de blocuri, ordonate pe categorii.

Fiecare categorie are o culoare. De exemplu, toate blocurile legate de mișcarea spritei sunt albastre.

- **Mișcare:** blocuri pentru a muta sprite (mișcare, rotire, ...) și variabile care se ocupă de poziția sau direcția lor
- **Arată:** blocuri pentru a modifica aspectul spritei. De asemenea conține blocuri care permit spritei să spună orice.
- **Sunet:** tot ce ține de sunet.
- **Evenimente:** blocuri pentru a reacționa la evenimente, în special atunci când utilizatorul face clic pe steagul verde.
- **Control:** blocuri care controlează execuția algoritmului, de ex. condiționate, bucle ...
- **Sensing:** blocuri care tratează interacțiunile cu utilizatorul, în a da o valoare.
- **Operatori:** blocuri pentru a face calculul (adăugare, multiplicare ...), pentru a alege un număr aleatoriu, concatenează două șiruri, condiții de calcul ...
- **Variabile:** permit crearea de noi variabile și liste. Acestea oferă, de asemenea, declarații pentru a le gestiona.
- **Blocurile mele:** acestea permit crearea de blocuri noi, care vor fi utilizate în proceduri definite.

Primul algoritm cu Scratch – un exemplu –

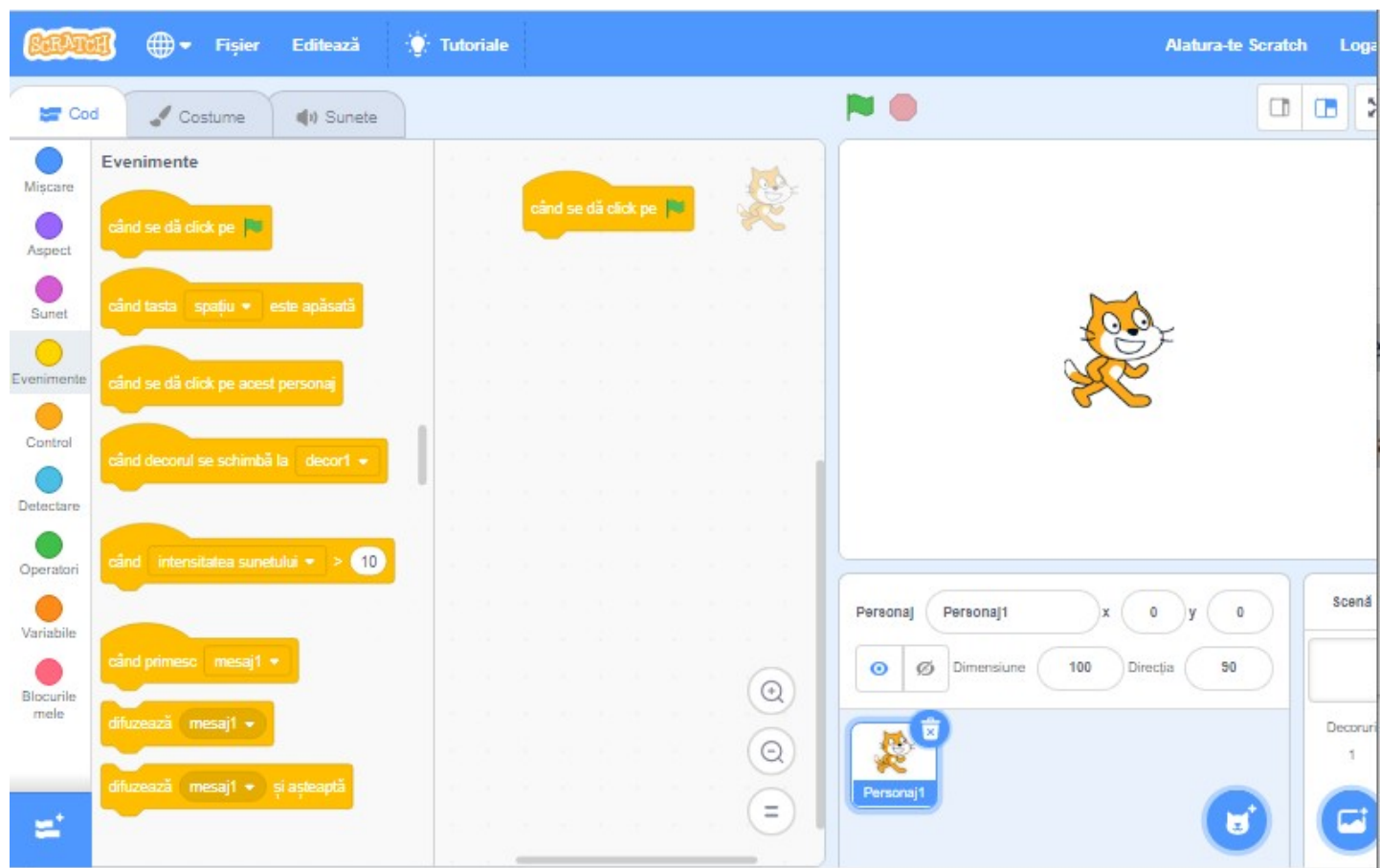
Introducere

Scop: • Executați 1 declarație când se face clic pe steagul verde.

Ca prima Activitate (A1) dorim atunci când utilizatorul face clic pe steagul verde, sprite-ul să se miște cu 10 pași de locația sa reală.

În primul rând, avem nevoie de un bloc care este activat atunci când evenimentul „faceți clic pe steagul verde” se declanșează. În bibliotecă, alegeți Evenimente și blocul.

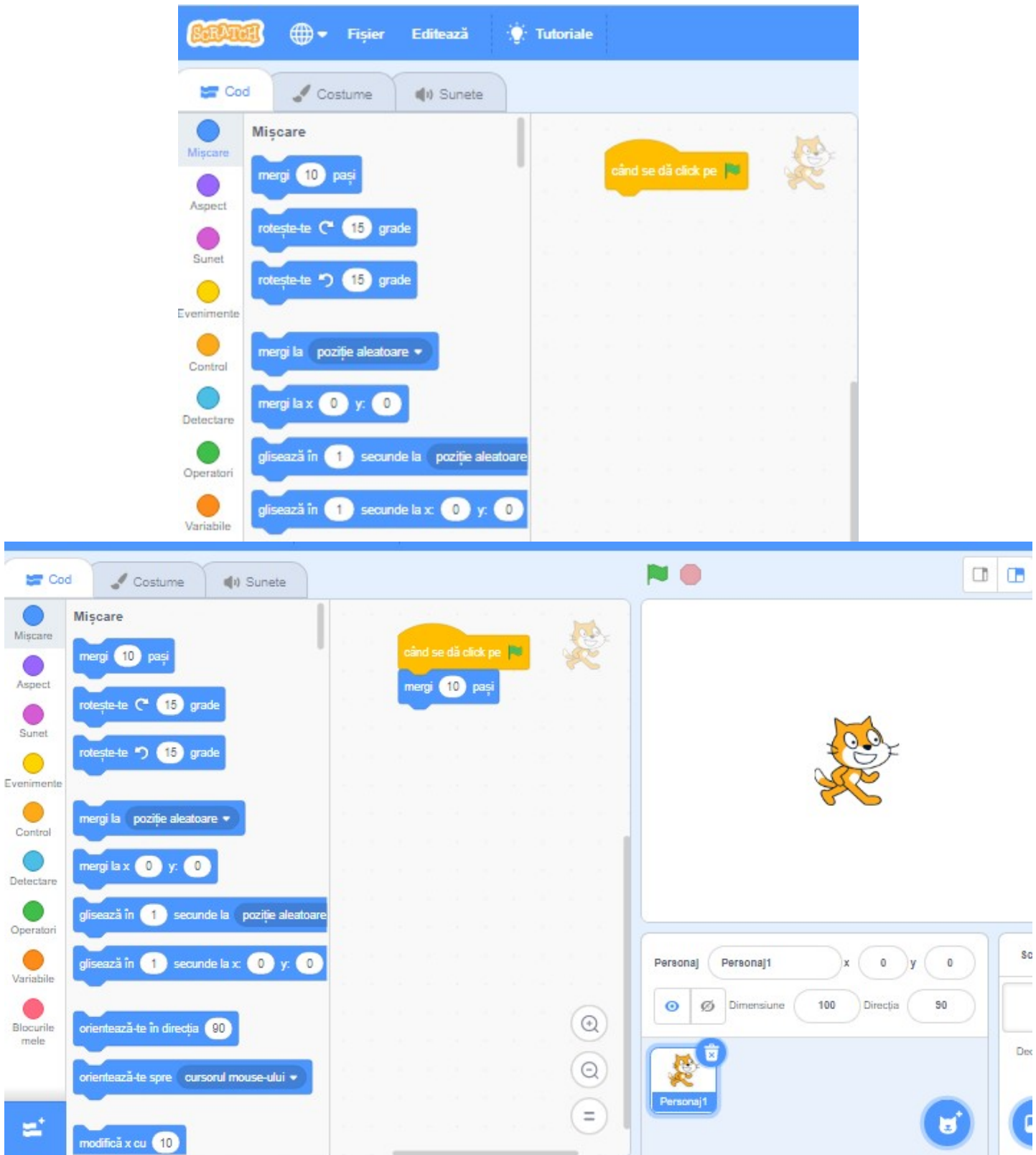
Glisați și fixați acest bloc în parte algoritm.



În bibliotecă, alegeți Evenimente și blocul. Glisați și fixați acest bloc oriunde în parte algoritm.

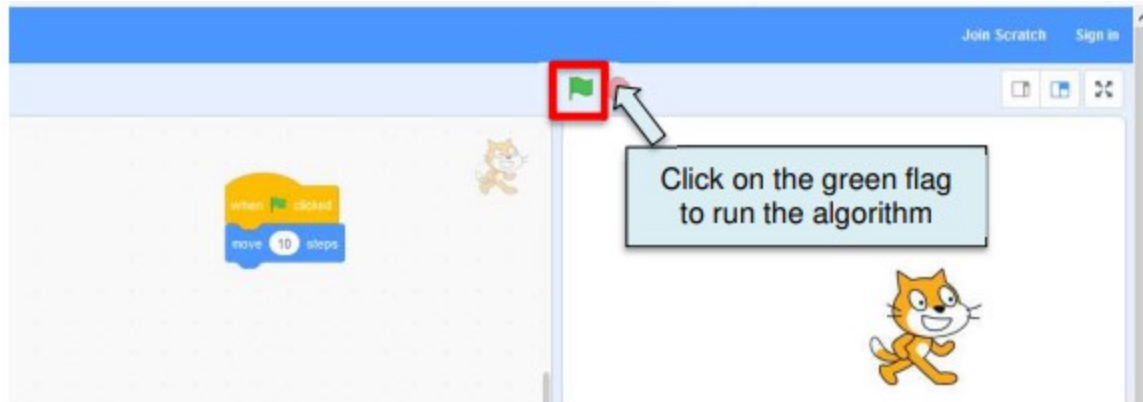
Blocul care descrie mișcarea spritei poate fi găsit în Set de declarații de mișcare. Alegeți, glisați și fixați-l în parte algoritm. Glisați al doilea bloc (mutați) chiar sub cel anterior, vor rămâne împreună.

Introducere



Click-and pe Drapelul Verde declanșează cele două instrucțiuni care se vor executa secvențial începând se sus în jos.

Introducere



Secvența instrucțiunilor

Așa cum este definit în prima secțiune, un algoritm este astfel o secvență de pași de calcul care transformă intrarea în ieșire. Putem defini fiecare dintre acești pași ca o declarație (de exemplu, declarație variabilă sau afectare, declarație buclă, declarație condițională ...).

Pentru a calcula ieșirea din intrare, sunt necesare în general mai multe instrucțiuni, că poate fi executat secvențial.

În special, sunt necesare secvențe de instrucțiuni pentru mai multe calcule, care stochează rezultatele intermediare în variabile, repetă de mai multe ori a același subset de instrucțiuni cu diferite valori variabile sau execuți două sau mai multe enunțuri alternative în funcție de rezultatul unui calcul anterior sau al unei interacțiuni cu utilizatorul.

Putem distinge două tipuri de declarații:

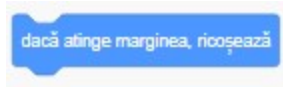
- **simple:** aceste sunt adesea o comandă imperativă, practic doar 1 linie de cod sau 1 bloc în Scratch.
- **încorporarea enunțurilor:** aceste enunțuri includ o succesiune de enunțuri. Este de exemplu cazul pentru bucle sau condiționare. Buclele vor executa fișierul secvența de afirmații anexată de mai multe ori.

Declarațiile condiționale vor fi doar executate în secvența de instrucțiuni anexată dacă este îndeplinită o condiție dată. Scopul este să combinați mai multe afirmații sau adăugați o buclă infinită, sau să manipulați diferitele meniuri și evenimente, sunete, control, etc.)

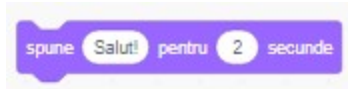


Un alt exemplu din algoritmul anterior (cu cele 2 blocuri Start și Mișcare), adăugați următoarele instrucțiuni:

Introducere



- dacă este pe margine, săriți:



- spune Bună ziua! timp de 2 secunde



- modificați dimensiunea

Indicație: culoarea blocului este similară cu culoarea din meniul bibliotecii.

Încercați: încercați să schimbați ordinea blocurilor și să vedeți ce se întâmplă

Exemplu 2: Cu algoritmul anterior, utilizatorul trebuie să facă clic de mai multe ori pe verde

steag pentru ca sprite-ul să se miște de mai multe ori.



Vom avea un bloc pentru a repeta pentru totdeauna 4 blocuri le-am adăugat deja. Adăugați astfel blocul pentru a repeta cele patru afirmații.



Notă: blocul este diferit de cele 4 blocuri anterioare pe care le-am folosit anterior ca încorporează un set de afirmații în sine.

Are un început și un sfârșit, între începe și termină un set de declarații poate fi inclus.



Introducere

Puteți astfel să glisați și să fixați în fișierul bloc sub bloc și în jurul celor 4 instrucțiuni.

Astfel, adăugați blocul pentru a repeta Încercați-l: încercați să scoateți câteva afirmații din bloc și să vedeți ce se întâmplă. Schimbați ce este în interiorul și în afara blocului.

Sugestie: în Scratch manipularea afirmațiilor inserate într-un algoritm poate părea a

puțin complicat. Rețineți că atunci când mutați o instrucțiune dintr-un bloc, aceasta ia cu ea toate

afirmații lipite sub ea.

Rețineți, de asemenea, că puteți pune afirmații sau bloc de afirmații pe care nu le utilizați

oriunde în fereastra algoritmului. Dacă nu sunt blocați sub, nu o vor face să fie luate în considerare la executarea codului.



Blocat sub insemna contrariu este:

În acest fel, puteți „pune” cu ușurință “blocuri sub” sau instrucțiuni utilizând în prezent sau / si refolosindu-le ulterior.

Variabilele

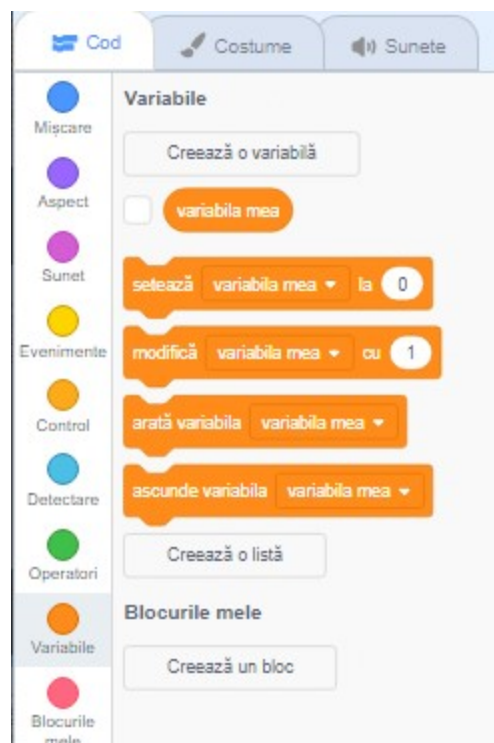
Într-un algoritm, este adesea util și necesar să stocăm rezultatele intermediare, calcule și să le putem reutiliza în continuarea algoritmului.

Se face folosind variabile. Timpul care se scurge de la începutul jocului, scorul care crește la un anumit eveniment, numărul de vieți care scade la un anumit eveniment,

Într-o variabilă, putem stoca o valoare numerică (întreg sau float), o valoare șir sau boolean (numai adevărat sau fals). Dar o variabilă poate stoca și un set (o listă) de elemente.

În orice algoritm, variabilele trebuie mai întâi declarate și inițializate, adică prima declarație într-un algoritm ar trebui să fie dedicată numirii și să atribuie o valoare inițială variabilei.

Este important să alegeți cu atenție numele variabilei, astfel încât să poată fi semnificativ pentru dezvoltator însuși, dar și pentru orice alt dezvoltator care ar putea citi și reutiliza algoritmul.



Introducere

Variabilă nouă

Numele variabilei:

☒ Pentru toate personajele ☐ Doar pentru acest personaj

Renunță OK

Variabilă nouă

Numele variabilei:

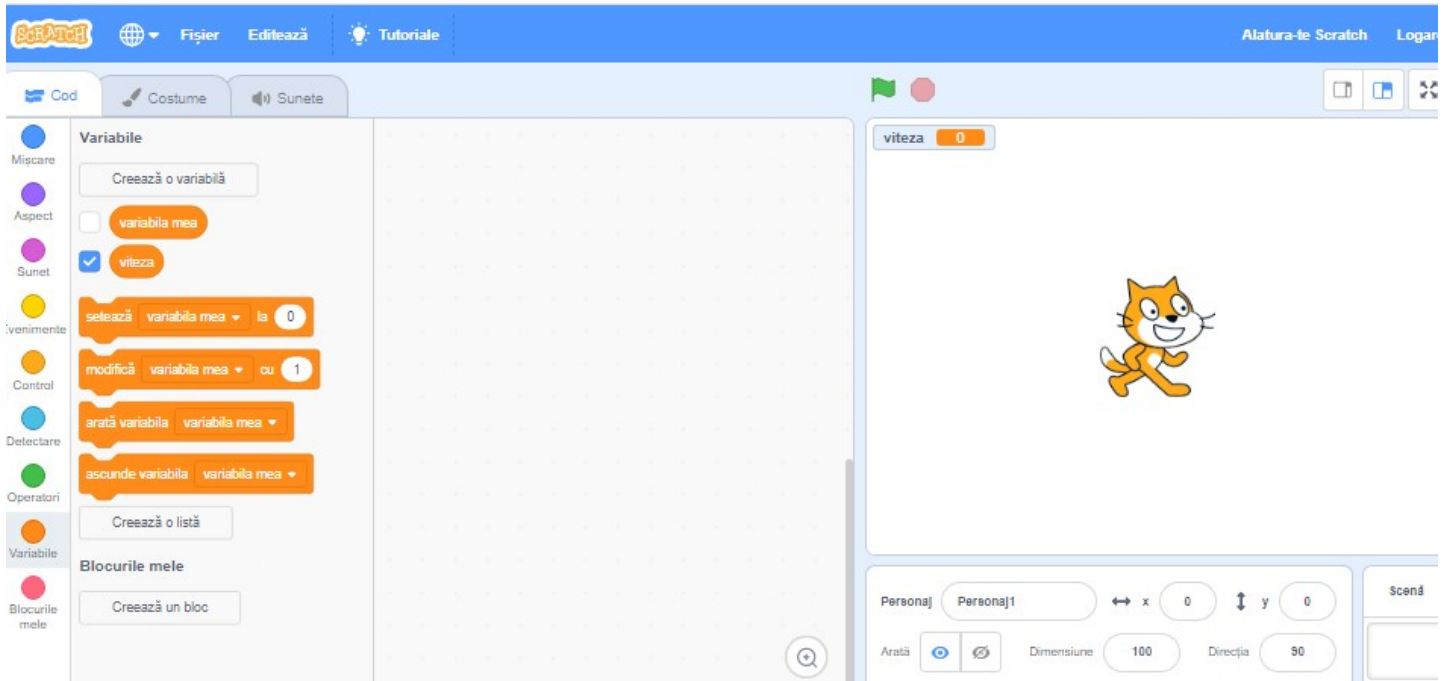
viteza|

☒ Pentru toate personajele ☐ Doar pentru acest personaj

Renunță OK

Nume Variabila

Introducere



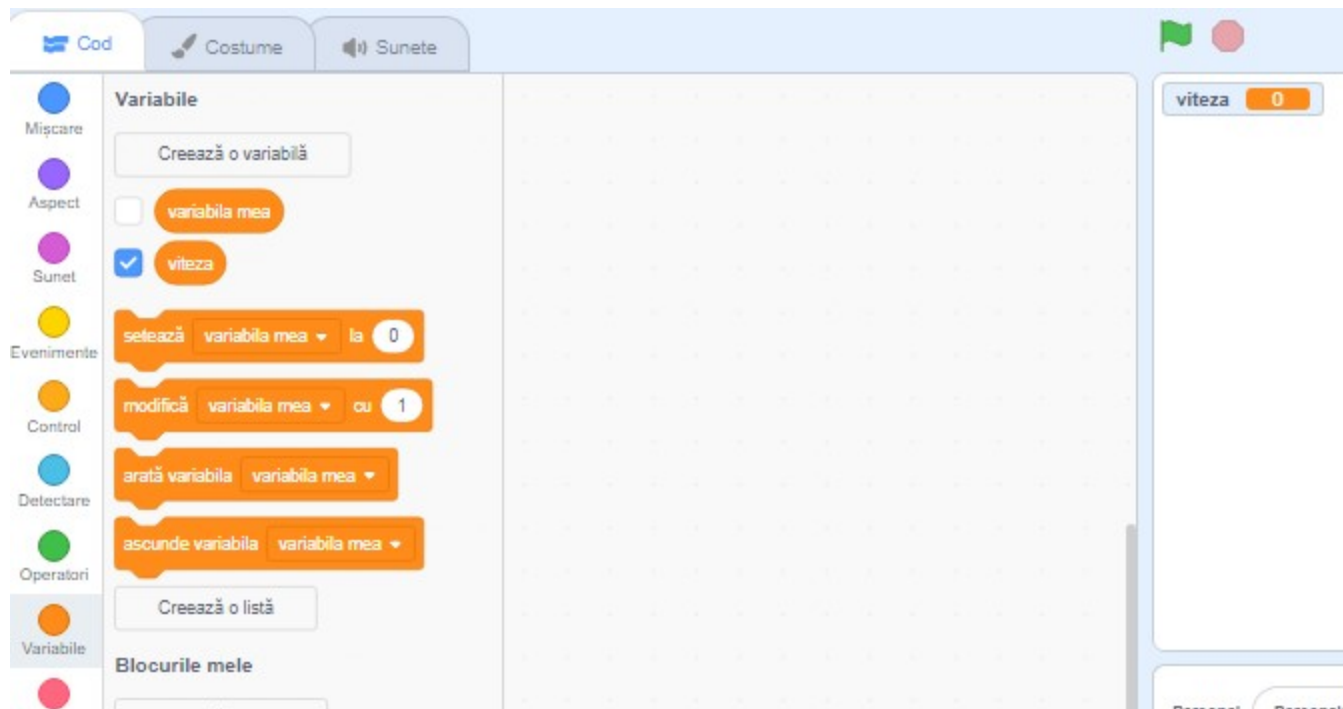
Variabile definite de utilizator: Scratch permite dezvoltatorului să creeze propriile variabile. Meniul Variabile este dedicat acestui scop. Pentru a crea o nouă variabilă, faceți clic pe

Butonul „Creați o variabilă”. Scratch întreabă numele variabilei.

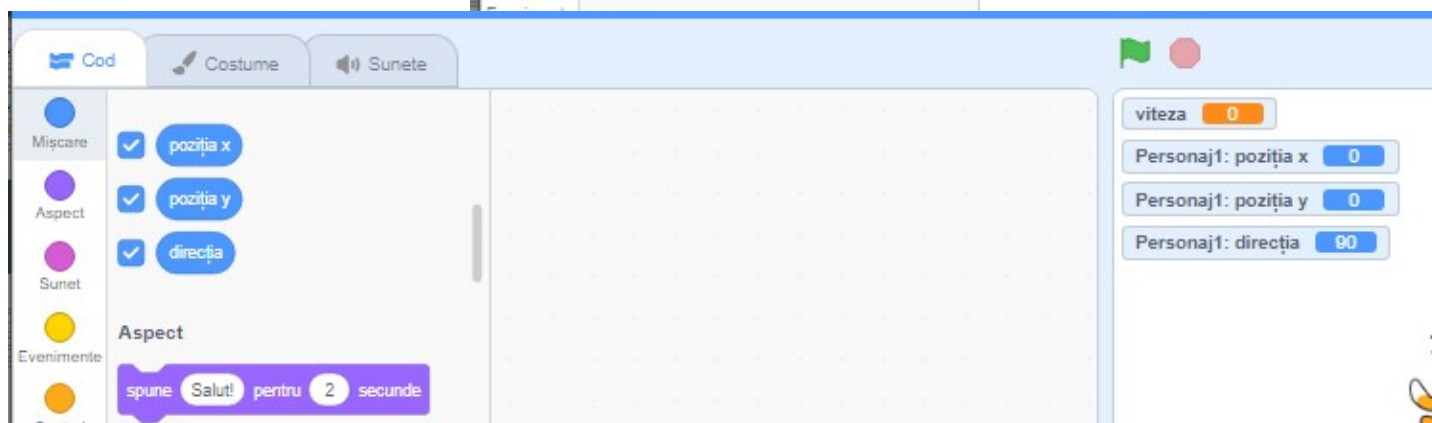
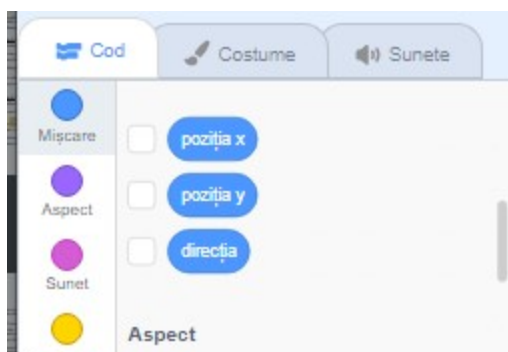
În captura de ecran, am ales „viteza”.

Scratch creează, de asemenea, un set de blocuri noi care permit gestionarea variabilei: schimbarea valorii variabilei, iar valorile variabilelor definite de utilizator pot fi vizualizate și modificate în partea stângă sus a colțului vedere de execuție.

Introducere

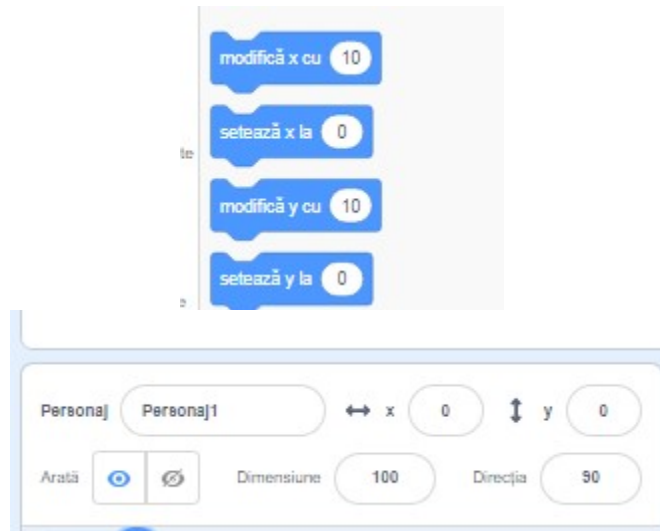


Variabile predefinite de Scratch la instalare sau incorporate (built-in). În Scratch, spritele au variabile încorporate, cum ar fi locația lor, sau dimensiunea forma lor. Aceste variabile pot fi vizualizate și modificate direct în sprite în panoul variabil, sub vizualizarea de execuție sau pot fi modificate în mod particular declarați (de exemplu).



Majoritatea blocurilor **Miscare** pot modifica aceste variabile.

Introducere



sau

Scopul lor este sa:

- Afișați poziția: manipulați operatorii pe variabilele șir incorporate (înlănțui)
- Măriți viteza spritei: manipulați operatorii pe un utilizator numeric definit variabil
- Schimbați 2 variabile.

Exemplu:

Folosind algoritmul din exemplu din secțiunea anterioară ca punct de pornire, modificați-l pentru a afișa coordonatele spritei:

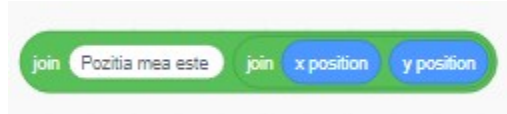


- Faceți sprite-ul să spună poziția x (în loc de Hello) cu



- Faceți spriteul să spună „Pozitia mea x este” și valoarea variabilei poziției x, folosind blocul pentru a concatena șiruri (funcționează pentru a concatena un șir și un număr).
- Spuneți spritei „Locația mea este: x = poziția x și y = poziția y” (în loc de poziția x și poziția y, ar trebui să afișeze poziția x curentă și poziția y variabilă), folosind mai multe blocuri.

Introducere

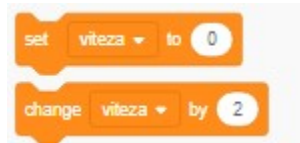


Un nou exemplu:

Să luăm în considerare algoritmul anterior. Sprite-ul se va deplasa pentru totdeauna la aceeași viteză (10).

Acum ne propunem să creștem viteza spritei după fiecare mișcare.

- Creați o nouă variabilă (pentru a reprezenta viteza spritei).
- Inițializați acea viteză la 0 la începutul algoritmului.
- Utilizați variabila de viteză în instrucțiunea de mutare.
- Măriți viteza după fiecare mișcare.



Folosiți

Alt exemplu: Creați un algoritm nou, cu 2 variabile x și y, inițializate la 10 și 50.

Scrieți algoritmul care schimbă valoarea celor două variabile x și y. La final, x ar trebui să fie egal cu 50 și y cu 10.

Sugestie: o variabilă suplimentară poate fi utilă.

Declaratii conditionate

Fluxul algoritmului ar trebui să se poată adapta diferitelor variabile posibile sau valori de intrare.

În acest scop, algoritmii folosesc instrucțiuni condiționale. Aceste instrucțiuni pot executa un set de instrucțiuni dacă și numai dacă o condiție este adevărată sau alegeți între două seturi de enunțuri în funcție de valoarea condiției.

Poate lua următoarele două forme:

Introducere

DACA (condiția este adevărată)

ATUNCI

executați un set de instrucțiuni

ÎNCHEI DACA



sau

DACA (condiția este adevărată)

ATUNCI

executați set de declarații1

ALTE

executați set de declarații2

ÎNCHEI DACA



Condiția ar putea fi de exemplu egalitatea între variabile ($x = y$), semnul a

variabilă ($x > 0$) sau orice altă expresie care poate fi evaluată fie la adevărat, fie la fals.

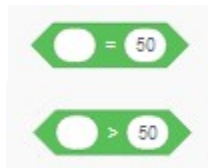
Introducere

Pentru exemplu în algoritmul din secțiunea anterioară, viteza ar putea fi crescută numai dacă rămâne sub o valoare maximă.

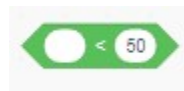
În Scratch, aceste două forme de instrucțiuni condiționale sunt (în panoul de control):



În fiecare dintre ele poate fi încorporat un set de afirmații după atunci și celălalt.



Pentru a scrie condițiile, Scratch oferă următorii operatori: sau sau



În plus, operatorii logici permit combinarea condițiilor:



(returnează inversul valorii condiției, adică adevărat dacă condiția este falsă și falsă dacă

condiția este adevărată), (returnează adevărat dacă și numai dacă ambele condiții sunt adevărate,

și fals în caz contrar).

De exemplu, condiția este adevărată dacă și numai dacă i este superioară la 50 și x mai mic decât y. În alte cazuri, este fals.

Declarații de buclă

În algoritmi, este adesea util să repetați de mai multe ori aceleași afirmații. La fiecare nou

apel al setului de enunțuri, numai valoarea unei variabile poate fi modificată. Evită scrierea

de mai multe ori aceleași linii din algoritm.

Devine, de asemenea, necesar să utilizați instrucțiuni de buclă atunci când numărul de repetări nu este

cunoscut când se scrie algoritmul.

Numărul de repetări poate fi o intrare a utilizator, ca într-un algoritm care calculează suma sau media primelor N numere întregi.

Sau poate depinde de starea unei variabile, de exemplu, atunci când un algoritm ar trebui

Sa calculeze suma elementelor setului sau a unei liste.

În cele din urmă, în mai multe cazuri, numărul de repetări nu este cunoscut la începutul buclei.

De exemplu, un algoritm poate cere utilizatorului să introducă un număr mai mare de 10

iar algoritmul repetă această cerere până când utilizatorul a tastat un număr corect.

Declarațiile de buclă pot lua următoarele forme:

LOOP (de N ori)

set de enunțuri

END LOOP

sau

LOOP PÂNĂ (condiția este adevărată)

set de enunțuri

ÎNCHEI BUCLA PÂNĂ

sau

LOOP WHILE (condiția este adevărată)

set de enunțuri

ÎNCHEIEȚI BUCLA CÂND

Prima formă este utilizată atunci când numărul de repetări este cunoscut când începe bucla.

Din contră, LOOP PÂNĂ și LOOP WHILE au un număr necunoscut de repetări.

LOOP PÂNĂ când se repetă până când condiția devine adevărată, în timp ce LOOP WHILE este repetată în timp ce condiția este adevărată.

În Scratch, cele două tipuri de instrucțiuni de buclă sunt:



• •

În Scratch există, de asemenea, un bloc specific pentru repetarea definitivă a instrucțiunilor încorporate:



Exemple:

Introducere

- Repetați de N ori;14
- Repetați până la bucla cu 1 condiție;
- Echivalența între repetarea de N ori și repetarea până;
- Repetați până la bucla cu mai multe condiții.

Exemplul 1:

Scrieți un algoritm care repetă de 10 ori următoarele afirmații:

- spriteul mișcă de ce 10 pași
- așteaptă 1 secundă.

Sugestie: așteptați se află în meniul Control

Exemplul 2:

Adăugați la algoritmul anterior faptul că sprite-ul spune „Pasul 1” după prima sa mutare,

„Pasul 2” după al doilea și așa mai departe ...

Sugestie: se poate crea o variabilă suplimentară pentru a stoca numărul de repetări.

Exemplul 3:

Scrieți un algoritm care mișcă spritul până când poziția sa x este mai mare sau egală cu 100.

Exemplu 4:

Scrieți un algoritm care calculează și afișează suma primelor 12 numere întregi cu 2

algoritmi alternativi, unul cu o buclă „repetă de N ori” și unul cu „repetă până”

buclă.

Introducere

Exemplu 5:

Scrieți un algoritm pentru a juca următorul joc cu sprite-ul.

Programul alege un număr aleatoriu între 1 și 20. Apoi îi cere utilizatorului să aleagă un număr și așteaptă un răspuns. Afișează (spune):

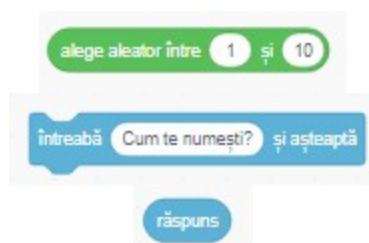
- „Prea scăzut! Încercați din nou ”: dacă utilizatorul a propus un număr mai mic decât cel ales
- număr
- „Prea sus! Încercați din nou ”: dacă utilizatorul a propus un număr mai mare decât cel ales
- număr
- „Câștigi! ”: Dacă utilizatorul a propus numărul ales.

Programul ar trebui să continue în timp ce utilizatorul nu găsește numărul potrivit

Vezi raspunsurile in Activitati

Sugestii: următoarele blocuri pot fi utile:

- : pentru a obține o valoare aleatorie între 1 și 20. Valoarea poate astfel, să fie stocate într-o variabilă care să fie păstrată de-a lungul algoritmului.
- afișează un mesaj și așteaptă ca utilizatorul să introducă a valoare. Răspunsul utilizatorului este stocat în variabilă.



NB: Folositi blocurile

Manipularea seturilor (sau listei) de valori

Până în prezent, variabilele pe care le-am creat pot stoca o singură valoare.

Poate fi utilă manipularea unui set de valori. Cele mai simple exemple în matematică sunt vectori sau matrici.

De exemplu, dacă vrem să creăm o aplicație ajutând un profesor să gestioneze notele elevilor săi,

ar putea fi util să gestionăm împreună toate notele unui student să poată, de exemplu, să-l calculeze pe al său nota medie.

În plus, numărul de note poate să fie diferit pentru doi studenți diferiți, după cum se poate când am ratat câteva examene. O listă are avantajul că numărul de elemente pe care le conține nu este fix

a priori.

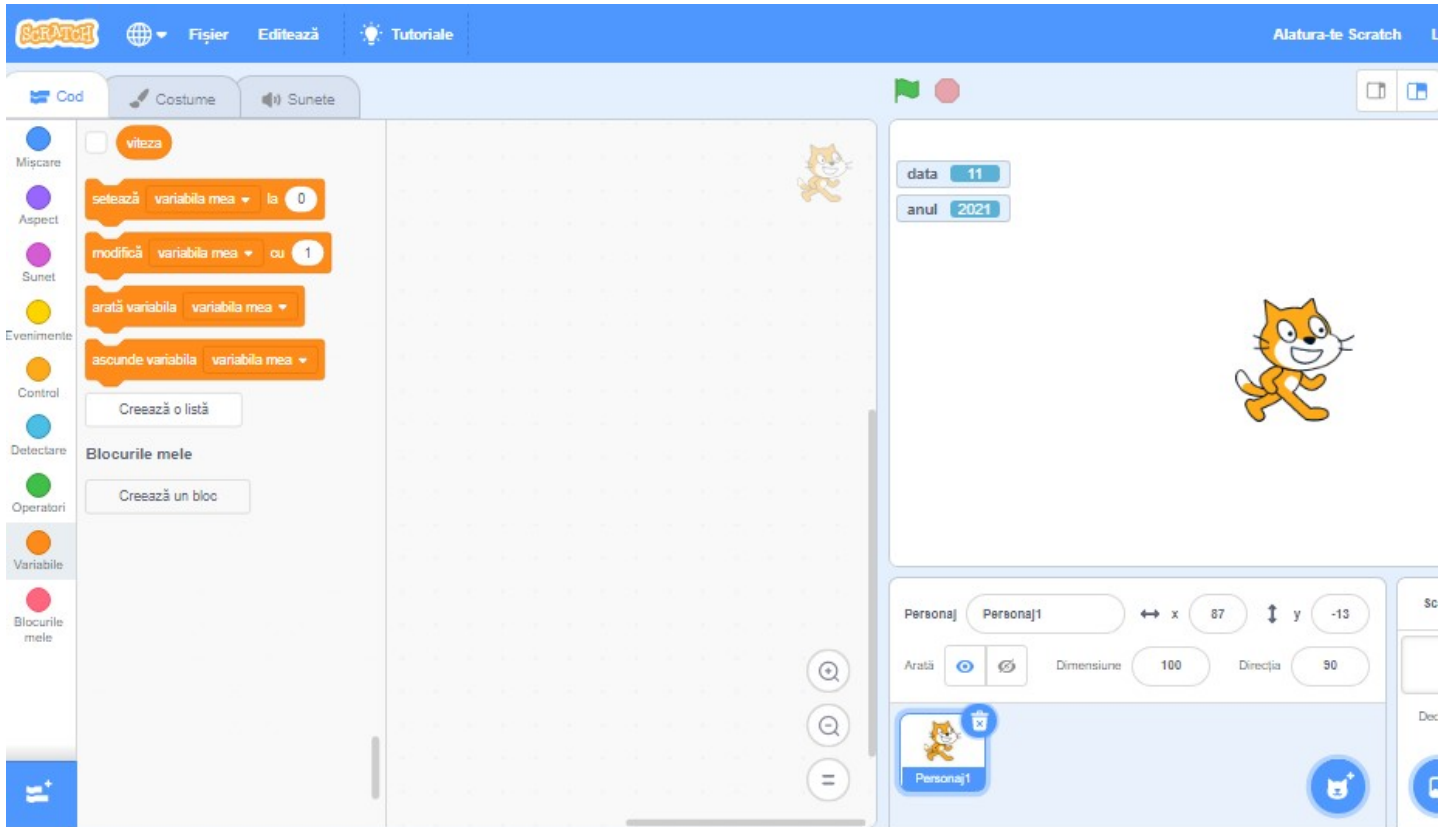
În general, o listă păstrează ordinea în care elemente au fost inserate. Într-o listă, pot fi adăugate elemente

(la sfârșit) sau inserat la o anumită locație (definit printr-o valoare a indexului întreg). Pot fi și ele eliminat sau înlocuit.

În cele din urmă, este posibil să obțineți al i-lea element al listei (fără a o elimina) sau pentru a obține lungimea listei (numărul de elemente).

În Scratch, în meniul Variable, putem crea și un listă (și dați-i un nume ales). Asta face disponibil un set de declarații pentru gestionarea listelor. Diferența între o variabilă și o variabilă de tip list

Introducere



Variabilă nouă

Numele variabilei:

listaMeal

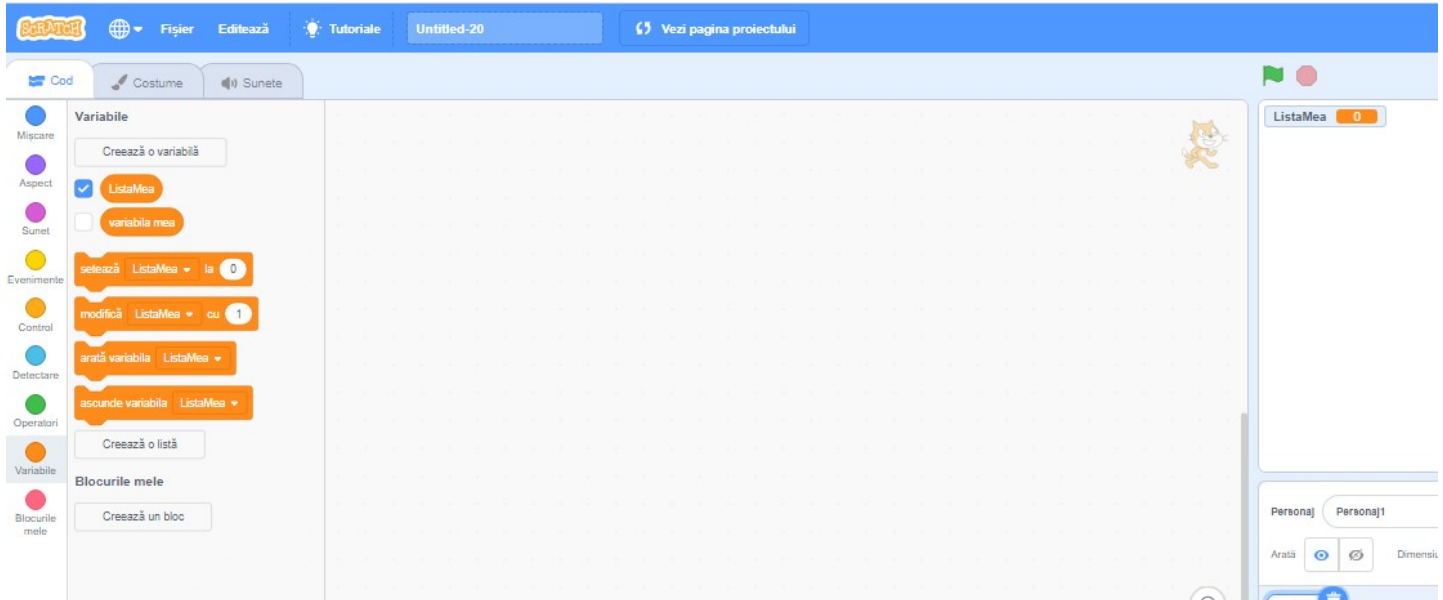
☒ Pentru toate personajele

☐ Doar pentru acest personaj

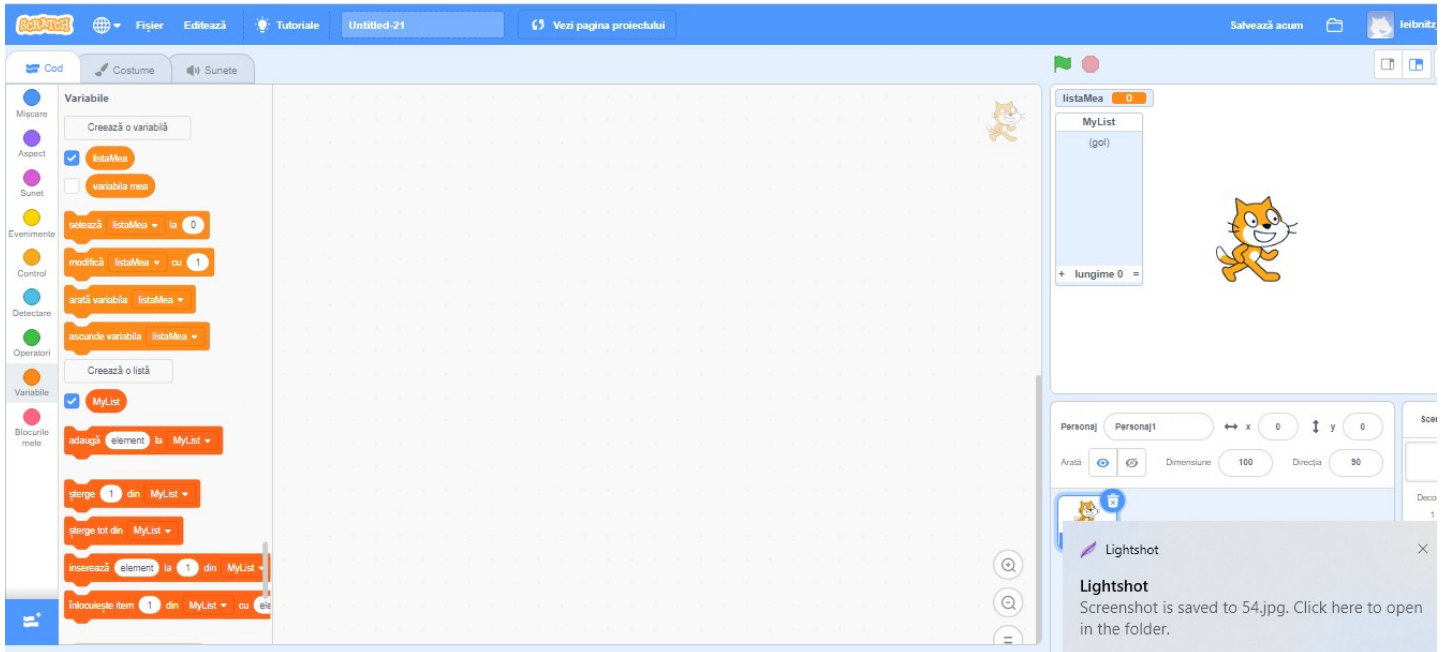
Renunță

OK

Introducere



Introducere



Obiective :

- Stocați datele într-o listă;

Introducere

Introducere

- Manipulați o listă;
- o Obțineți elementul maxim;
- o Sortează-l.

Exemplul 1:

Considerăm algoritmul Exemplul 3 al secțiunii Afirmații condiționale ca punctul de pornire (sprite-ul crește și scade viteza de la 0 la viteza maximă).

Ne propunem să stocăm toate pozițiile x în care sprite-ul a atins viteza maximă:

- Creați o nouă variabilă de listă
- La inițializare, golii lista (adică ștergeți tot elementul variabilei listei)
- Adăugați la listă locația x de fiecare dată când sprite-ul atinge viteza maximă. 16

** Exemplul 2:

Modificați algoritmul anterior pentru ca sprite-ul să spună întotdeauna x -ul maxim al listă (adică maximumul tuturor x la care a atins viteza maximă)

Sugestie: o nouă variabilă ar putea fi utilă. Această variabilă poate fi calculată de fiecare dată când viteza maximă este atinsă și un element adăugat la listă.

** Exemplul 3:

Modificați algoritmul Exercițiului 1, pentru ca lista să fie întotdeauna sortată. Acest lucru înseamnă că de fiecare dată când se adaugă o valoare, nu trebuie adăugată la final, ci inserată la locație care menține lista sortată.

Exemplul 4: (Joc de memorie).

Introducere

Sprite vă va cere să memorați o listă de numere care crește la fiecare pas.

Algoritmul va alege aleatoriu un număr între 1 și 100 și îl va stoca într-o listă de numere.

Acesta va spune utilizatorului noul număr și îi va cere să-l memoreze. Apoi întreabă

utilizatorul să introducă rând pe rând fiecare număr al listei. Când a cerut tot numărul de

lista, va alege un nou număr aleatoriu și așa mai departe ...

Jocul se termină de îndată ce utilizatorul tastează un răspuns greșit.

Exemplu de flux de joc:

Sprite spune: „Noul număr este: 35”. (așteptați 2 secunde)

Sprite spune: „Care este numărul 1?”

Tip de utilizator: „35”

Sprite spune: „Noul număr este: 47”. (așteptați 2 secunde)

Sprite spune: „Care este numărul 1?”

Tipuri de utilizatori: „35”

Sprite spune: „Care este numărul 2?”

Tipuri de utilizatori: „47”

Sprite spune: „Noul număr este: 2”. (așteptați 2 secunde)

Sprite spune: „Care este numărul 1?”

Tipuri de utilizatori: „35”

Sprite spune: „Care este numărul 2?”

Tipuri de utilizatori: „47”

Sprite spune: „Care este numărul 3?”

Tipuri de utilizatori: „3”

Sprite spune: „Pierzi. Joc încheiat”

Subalgoritm: procedură și funcție

Când algoritmii devin lungi în ceea ce privește numărul de linii și enunțuri, devine util pentru a păstra algoritmul clar și lizibil, să împărți un singur algoritm mai multe părți (care pot fi proceduri sau funcții sau myblocks in Scratch in engleza).

Procedurile sau funcțiile pot fi, de asemenea, utile atunci când se utilizează aceeași secvență de instrucțiuni în mai multe locuri din algoritm.

Apoi devine util să descompunem această parte într-un procedură sau funcție.

Procedurile sau funcțiile pot fi definite ca subalgoritmi care iau o oarecare intrare și folosiți-le în calcul pentru a produce un rezultat. O funcție este un subalgoritm care ia unele intrări și returnează un rezultat. Intrările nu sunt modificate.

Dimpotrivă, o procedură este un subalgoritm care nu returnează niciun rezultat, dar modificați valoarea intrărilor. O procedură este utilizată atunci când un subalgoritm trebuie să producă mai multe ieșiri.

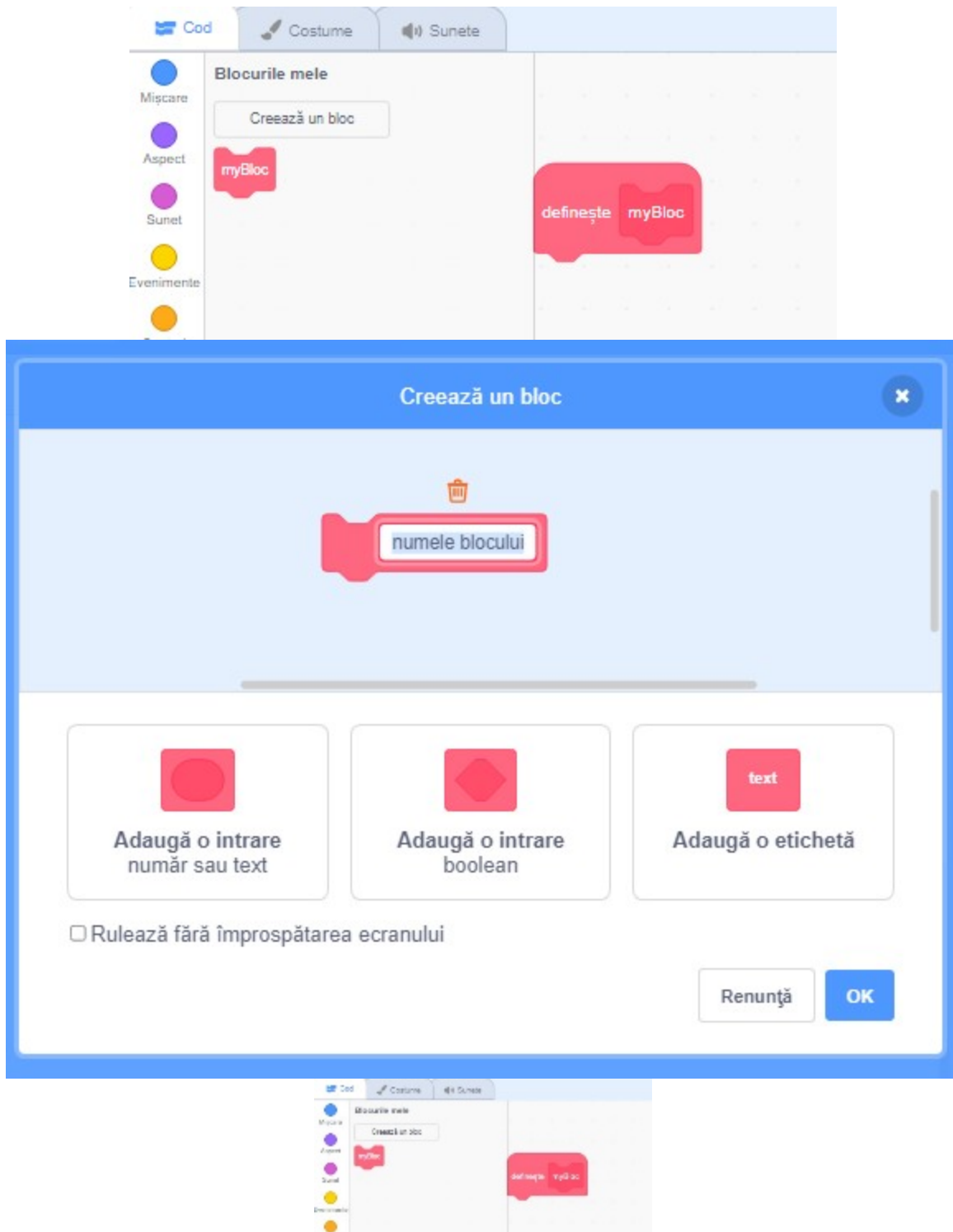
Scratch permite utilizatorului să definească un nou blocuri pentru definirea subalgoritmilor. Este

limitat la acest punct, deoarece un nou blocul nu poate returna nici o valoare și nici modificați parametrii de intrare. Va fi folosit pentru a modifica direct variabilele de spiritul.

Meniul „Blocurile mele” permite utilizatorului să definească un bloc nou, să modificați numele său și adăugați intrări (din tipuri variate).

Crearea unui nou bloc

Introducere



Meniul „Blocurile mele” permite utilizatorului să definească un bloc nou, să modifice numele său și să adăuge intrări (din tipuri variate).

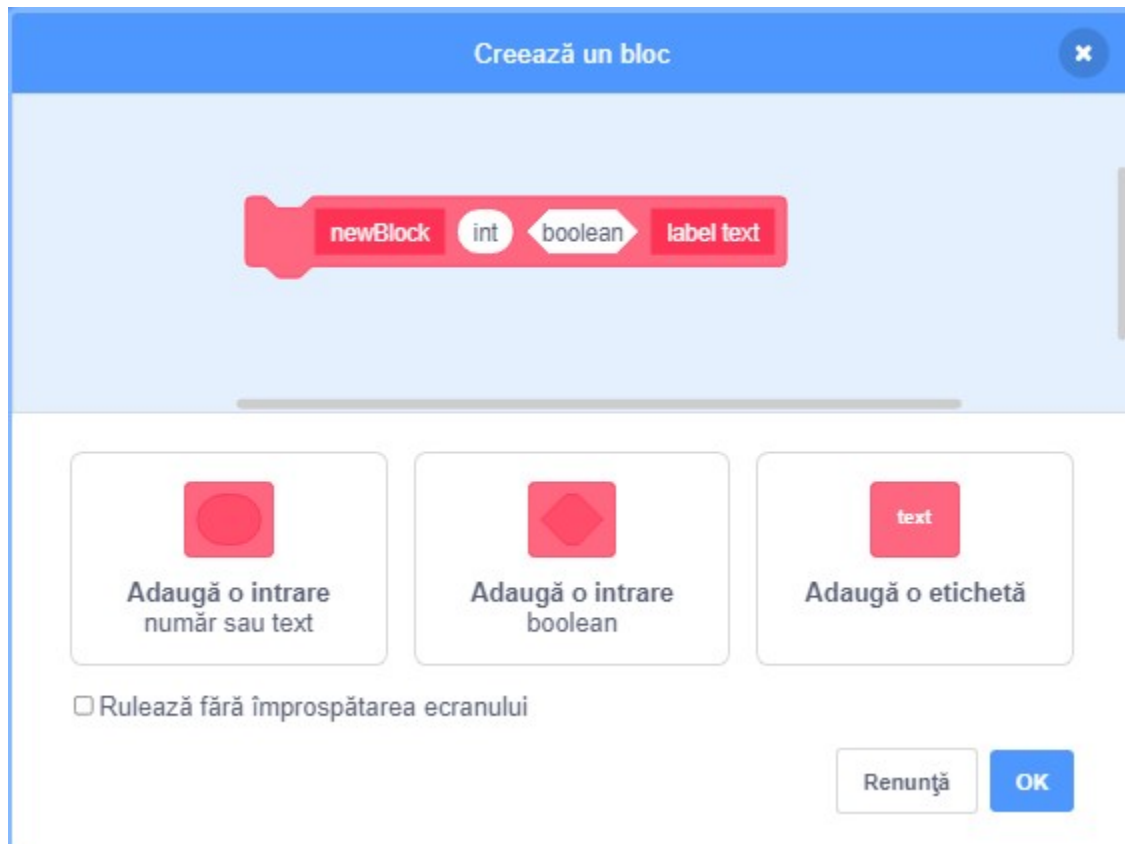
Crearea unui nou bloc creează un nou început de algoritm și un nou bloc permițând apelarea acestui subalgoritm oriunde în alți algoritmi. În exemplul de mai jos, `my_new_block` este un nou subalgoritm, în care `bool1`, `n1` și `text1` sunt cele 3 intrări

Introducere

parametrii.

NB: bool1, n1 și text1 devin, de asemenea, variabile care pot (și ar trebui) utilizate

numai în subalgoritm



Obiective:

- Definiți intrarea / ieșirea unei proceduri;
- Scrieți o procedură;
- Apelați o procedură;
- Ideea despre funcții.

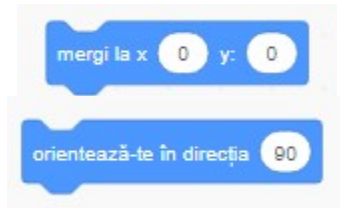
Exemplu 1:

Introducere

Scopul acestui algoritm este de a asculta clicurile mouse-ului pe ecran, pentru a obține coordonatele mouse-ului și pentru a localiza sprite-ul pe locația mouse-ului, direcția acestuia fiind în direcție 90.

În acest scop, definiți mai întâi un bloc nou MoveSprite, cu 2 intrări numerice (new_x și nou_y).

Acesta va localiza sprite-ul pe aceste noi coordonate și îl va indica în direcția 90.



Pot fi utile următoarele două blocuri: .

Vom scrie acum partea principală a algoritmului. Detalii despre algoritm:

- Așteptați până când utilizatorul a dat clic



- mutați sprite-ul la locația mouse-ului (apelând noul bloc MoveSprite)

- Apoi repetați pentru totdeauna:

- deplasați sprite-ul cu 10 pași și, dacă este pe margine, săriți
- dacă utilizatorul a dat clic undeva, mutați sprite-ul în această locație.

Sugestii:

- pentru a crea intrări pentru un bloc, faceți clic pe meniul „opțiuni” când îl creați sau, după ce acesta a fost deja creat, faceți clic dreapta pe el, selectați „editați” și faceți clic pe „opțiuni” meniul

- la definirea unui bloc în fereastra algoritmului, intrările sale pot fi preluate din definiția blocului și glisată ca variabile în cadrul instrucțiunilor

- Blocarea dacă utilizatorul a dat clic.

- Dacă utilizatorul a făcut clic, cele două variabile și conține coordonatele clicului.

**** Exemplul 2: (Calcul factorial)**

Scopul acestui algoritm este de a calcula și stoca într-o listă factorialul întregilor din

1 la 10 (sau orice N ...). Rețineți că factorialul unui număr întreg pozitiv este definit de: $n! =$

$1 * 2 * 3 * \dots * (n-1) * n$. De exemplu: $4! = 1 * 2 * 3 * 4 = 24$.

- Mai întâi creați o variabilă fact și o variabilă listă.

- Creați un bloc nou care calculează factorialul unui număr dat în intrare. Rezultatul este stocat în variabila de fapt.

- Creați algoritmul principal care calculează, pentru fiecare i de la 1 la 10, valoarea lui i!

și adăugați-l în listă. Lista conține astfel la fiecare index i valoarea lui i !.

Către o abordare orientată obiect / agent

Avem acum o imagine de ansamblu globală a principalelor principii și structuri ale algoritmilor.

În algoritmii bazați pe agenți (și orientați pe obiecte), se utilizează toate aceste principii.

Dar în plus, algoritmul este conceput luând în considerare entitățile care compun fenomenul

a modela. Fiecare tip de agent (sau obiect) are propriile variabile care descriu starea sa și propria sa

algoritmi (funcții și proceduri) care descriu comportamentele sale.

De exemplu, toate „mașinile” obiectele sau agenții pot fi descriși cu variabile precum „culoare” sau „marcă” și au

Introducere

comportamente precum „turn_wheel” sau „change_break”,

..... în timp ce toate obiectele sau agenții „pisică” pot fi descrise prin variabile precum „vârstă” sau „preferințe” și pot avea astfel de comportamente ca „get_older”, „mutare” sau „mâncare”.

Toți agenții sau obiectele de același tip vor avea aceleași variabile și aceiași algoritmi descriindu-le comportamentele, dar nu vor acționa identic, așa cum vor face variabilele lor de stare să fie distinct pentru fiecare agent sau obiect.

De exemplu: este posibil ca

- 1. 2 mașini să nu aibă aceeași marcă, și în funcție de marcă, comportamentul „change_break” poate acționa diferit;
- 2. 2 pisici s-ar putea să nu aibă aceeași vârstă și, în funcție de vârsta lor, comportamentul de „mișcare” poate acționa

diferit.

O idee despre obiecte / agenți poate fi găsită în Scratch cu sprites: puteți define sprite diferite și li se oferă variabile și algoritm diferit, astfel încât acestea să acționeze diferit atunci când apăsați steagul verde. Pisica face miou iar cainele latra, fiind vorba de 2 sprite diferite al categoriei animale

Cu toate acestea, nu este posibil să se definească tipurile de sprites așa cum este posibil să se facă atunci când se utilizează programarea orientată pe obiecte.

Esența programării orientate pe obiecte și a modelelor bazate pe agenți este de a define stări și comportamente ale agenților / obiectelor și apoi se gestionează modul în care pot interacționa cu fiecare și cu alții.

Acest lucru se poate face prin definirea comportamentelor agenților / obiectelor cum reacționează când percep ceva de la un alt agent / obiecte și / sau definesc comportamente care trimite mesaje altora.

Un alt nivel de control se face în general printr-un „programator” care controlează central atunci când un agent / obiect ar trebui să declanșeze un comportament.

În Scratch, spritele pot percepe distanța față de alți sprite și pot interacționa prin difuzare de mesaje și reacționarea la mesajele pe care le primesc.

Introducere

Dar aceste mesaje nu pot fi direcționate la un anumit sprite și nu există o altă modalitate prin care sprites să interacționeze.

De asemenea, nu există simțul unui „planificator” global, toate comportamentele sritelor fiind declanșate de evenimente.

Toate aceste concepte de agenți / obiecte și planificator vor face parte din cursul de instruire avansată SCARCTH , deci nu am inclus exerciții despre acest lucru în acest document.

Cu toate acestea, dacă tu sunt deja curios, verificați de exemplu în proiectul următor puteți vedea cum 2

spriturile pot fi definite și interacționează între ele <https://scratch.mit.edu/projects/569232558/editor>

EXERCITII SI REZOLVARI

rezolvarea lor, prin construirea unor algoritmi de

prelucrare a informației - conform cu PROGRAMEI SCOLARE DE CLASA aVa.

Nb: Gasiti pentru fiecare activitate si un clip video pe youtube, un link pentru codul sursa in scratch format sb3, si la cerere prin email la admin@itcool.online – ebook si flipbook + ppt de prezentare

NB: Notatie A1 – Activitatea 1Ai – Activitatea i

Nivelul 1

1. A1 Înțelegeți blocurile „mişcare 1”
2. A2 Înțelegeți blocurile „mişcare 2
3. A3 Înțelegerea blocurilor „punct de direcție”
4. A4 Înțelegerea blocurilor „punct în direcție 2”
5. A5 Mutați-vă în zigzag.
6. A6 Să-l mutăm pătrat.
7. A7 Animații care se mișcă la stânga și la dreapta.
8. A8 Intr-un unghi.
9. A9 Animație de mers pe jos.
10. A10 Mai multe costume
11. A11 Mai multe costume 2.
12. A12 Miau și să-l mutăm.

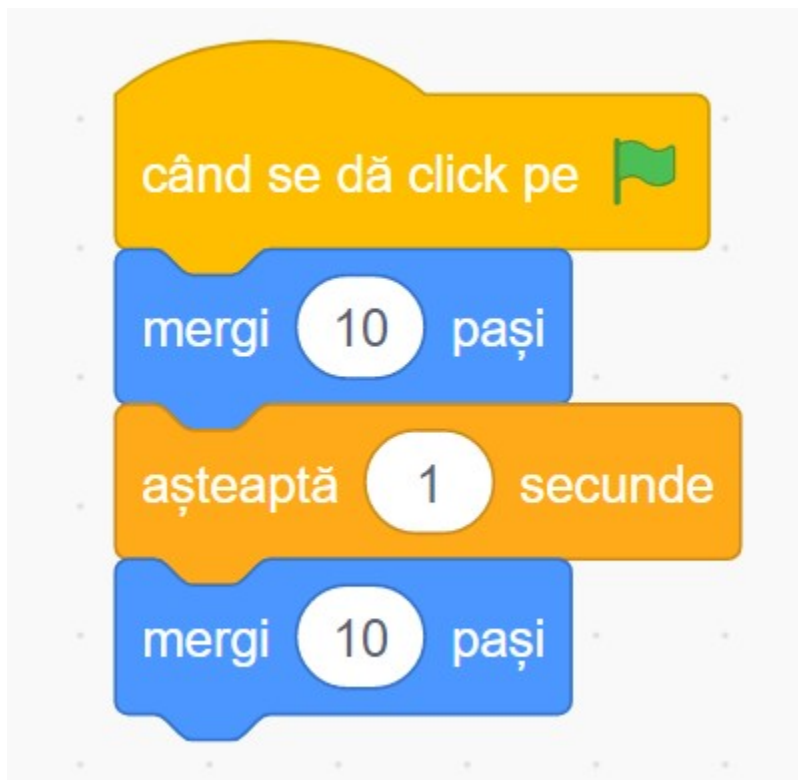
Nivelul 2

1. A13 Să-l mutăm în formă de cerc.
2. A14 Utilizați tastele săgeată pentru a o muta.
3. A15 Faceți clic pentru a muta sprite-ul.
4. A16 Folosiți o varietate de taste (introducere).
5. A17 Să schimbăm dimensiunea.
6. A18 Să-l facem mai mare mișcându-l.
7. A19 Să-l micșorăm mutându-l.
8. A20 Intermitent și schimbare costum.
9. A21 Alegeți costumul direct.
10. A22 Repetare cu un număr fix de ori.
11. A23 Să desenăm un dreptunghi cu un stilou.
12. A24 Să desenăm un cerc cu un stilou.
13. A25 Să desenăm un triunghi cu un stilou.

A1 Înțelegeți blocurile „mișcare”

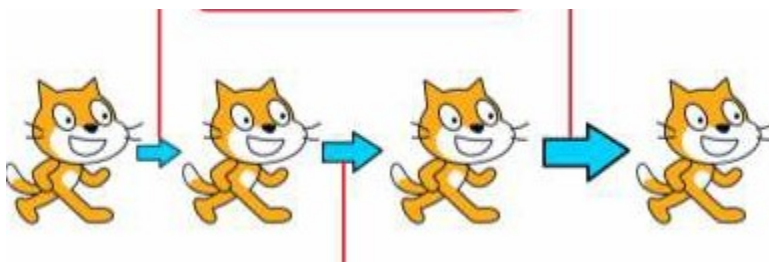
1. Incepeti programul click pe drapelul rosu
2. Miscati pisica Scratch 50 pasi
3. Miscati pisica Scratch dupa 1 secunda





A2 Intelege Miscarea Sprite 2

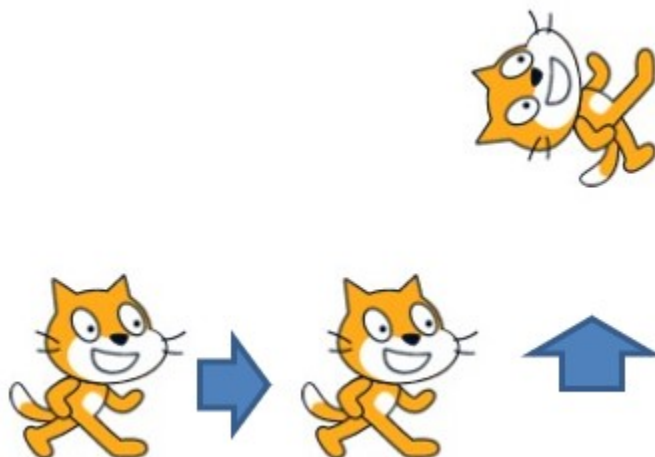
1. Incepeti programul click pe drapelul rosu
2. Miscati pisica Scratch 50 pasi
3. Miscati pisica Scratch dupa 1 secunda
4. Miscati pisica Scratch 50 pasi
5. Miscati pisica Scratch dupa 1 secunda
6. etc





A3 Înțelegerea blocurilor „punct de direcție”

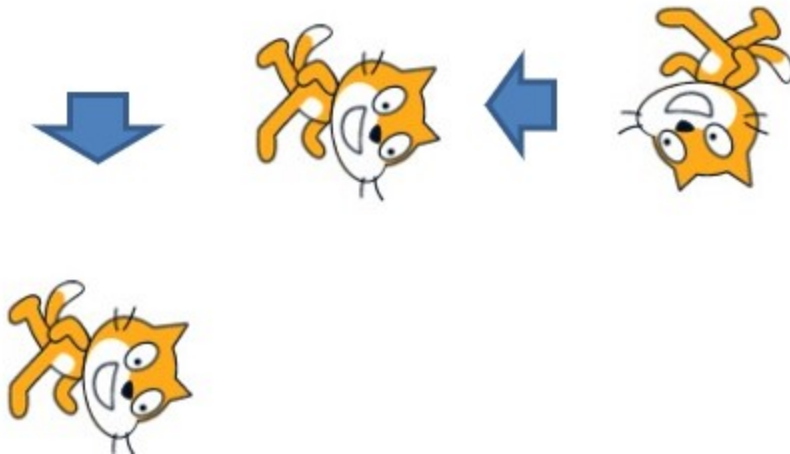
1. Start program click Drapelul Verde
2. Muta pisica 30 pasi
3. Muta 60 pasi dupa o secunda
4. Muta 90 pasi dupa 1 secunda



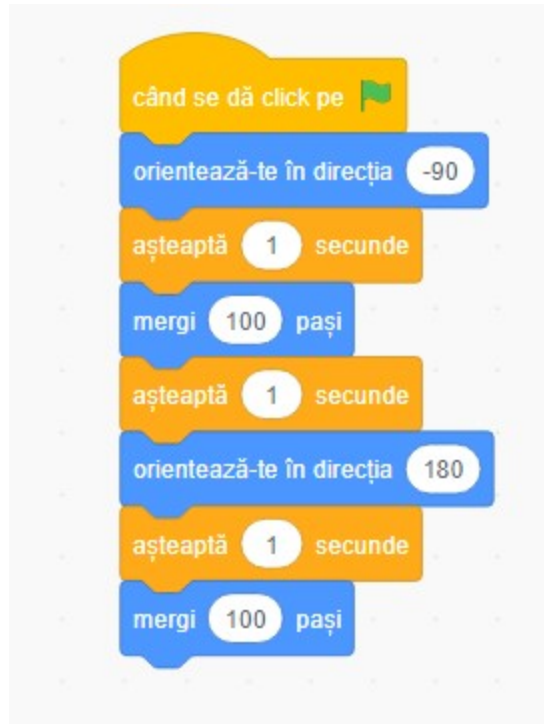
Introducere



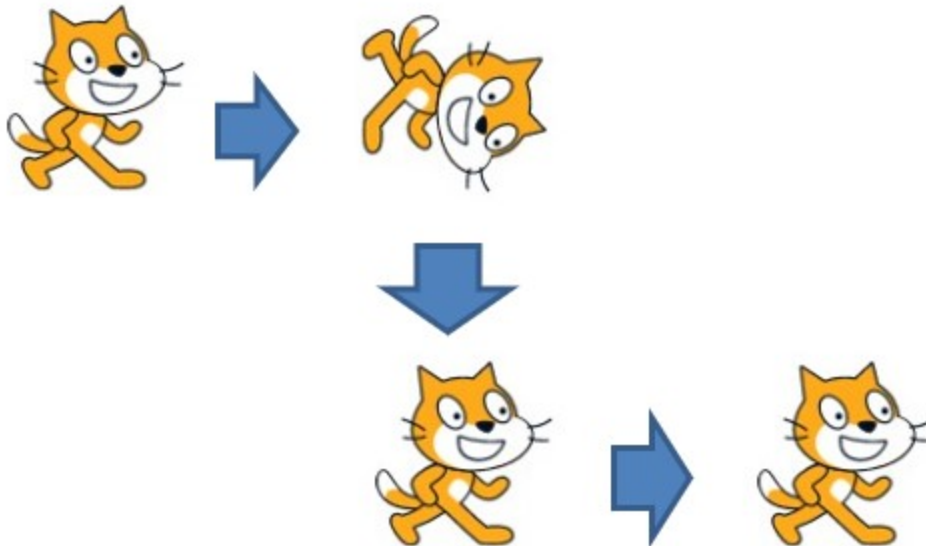
A4 Înțelegerea blocurilor „punct în direcție 2”



Introducere



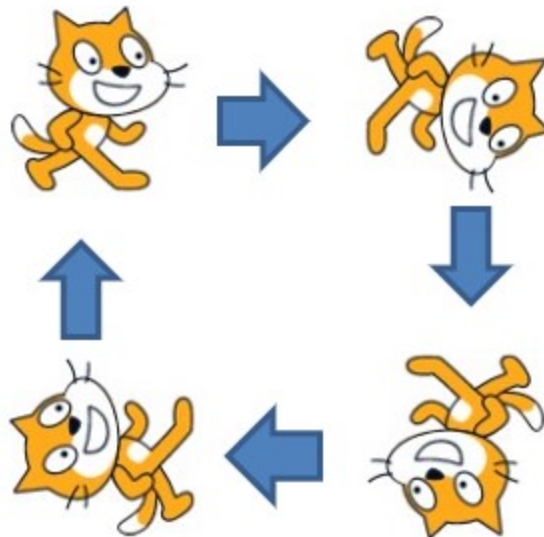
A5 Miscare Zig Zag



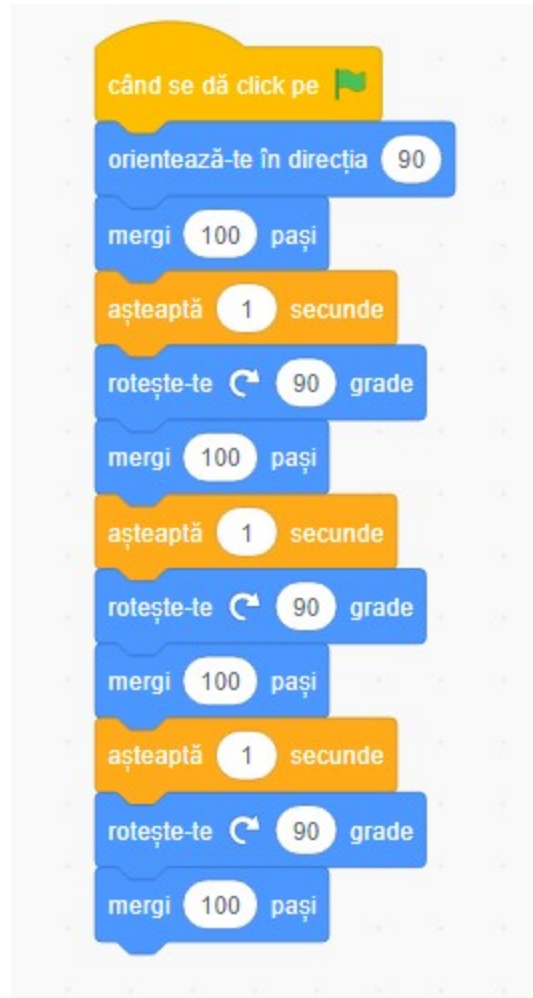
Introducere



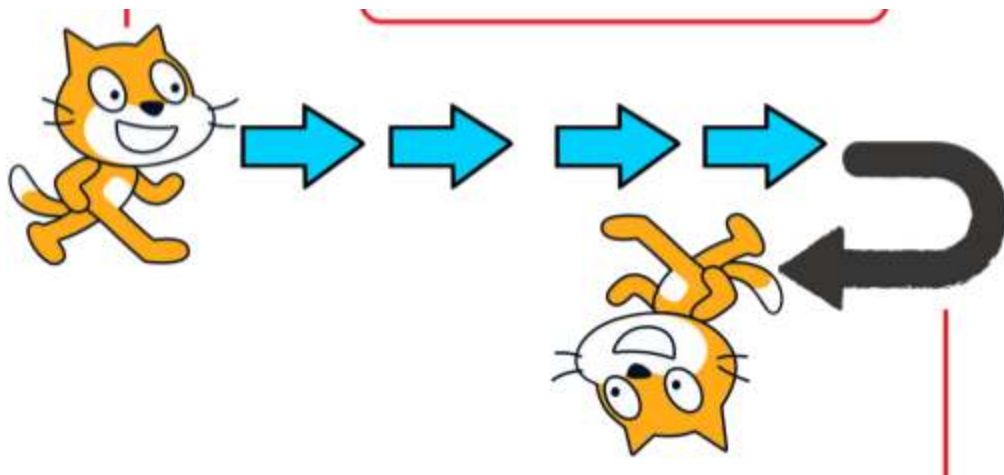
A6 Miscare in Patrat



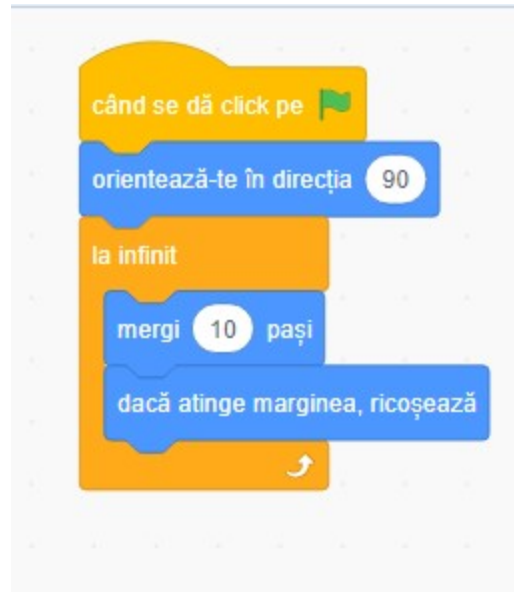
Introducere



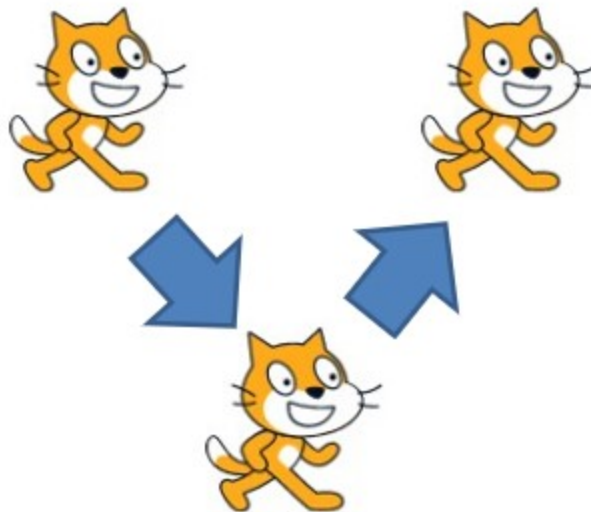
7 Animație cu miscare staga dreapta



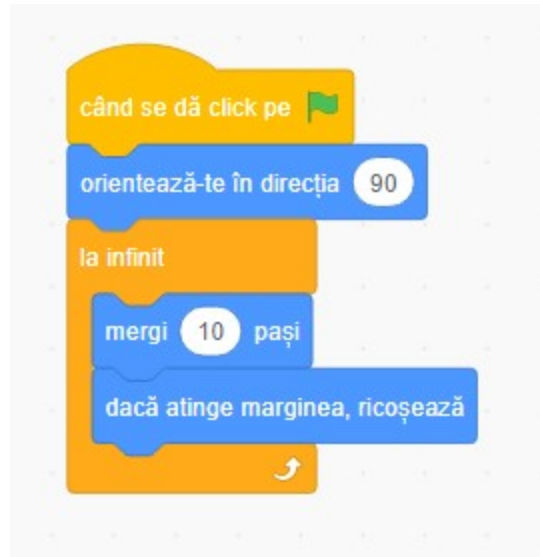
Introducere



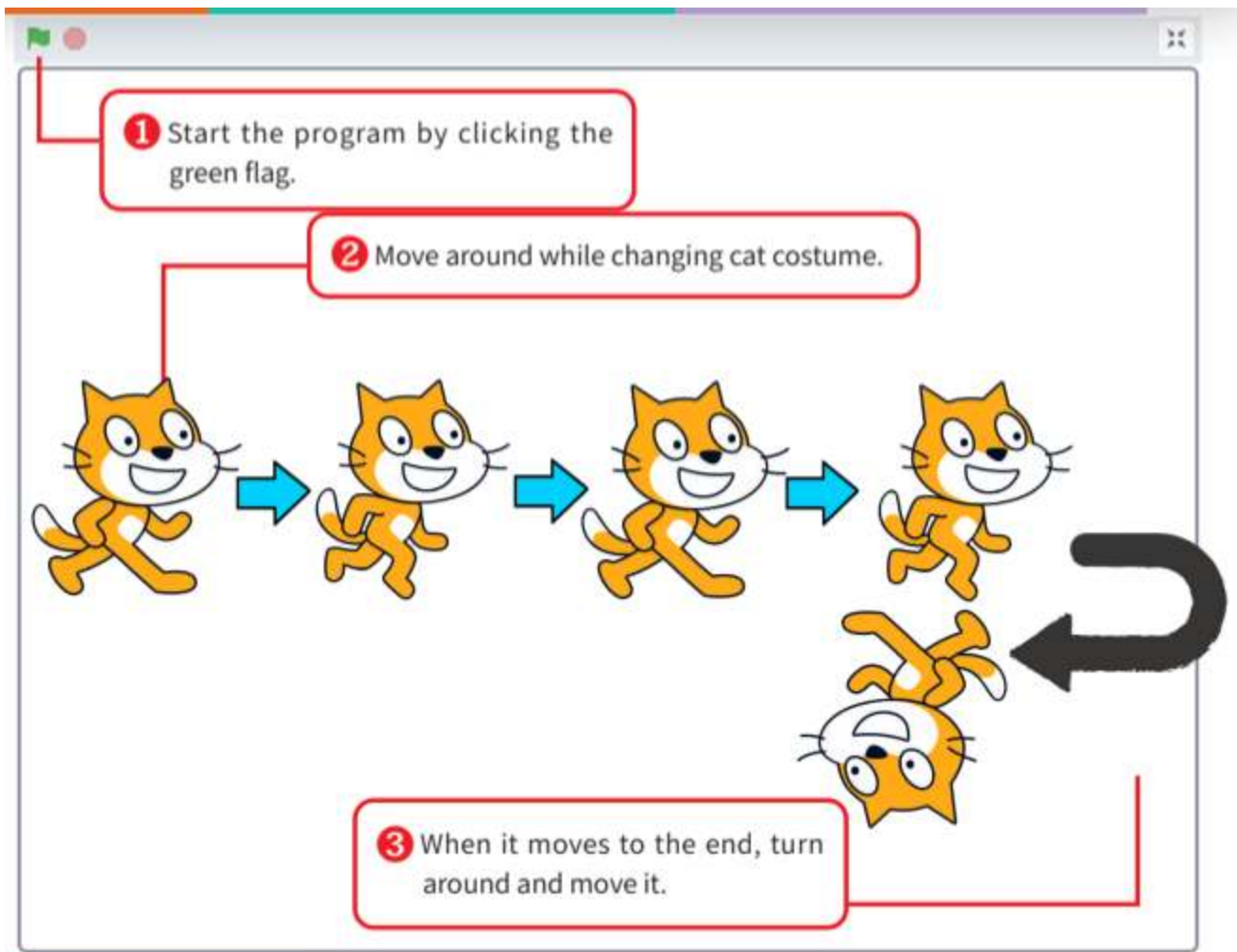
A8 Animatie in unghi



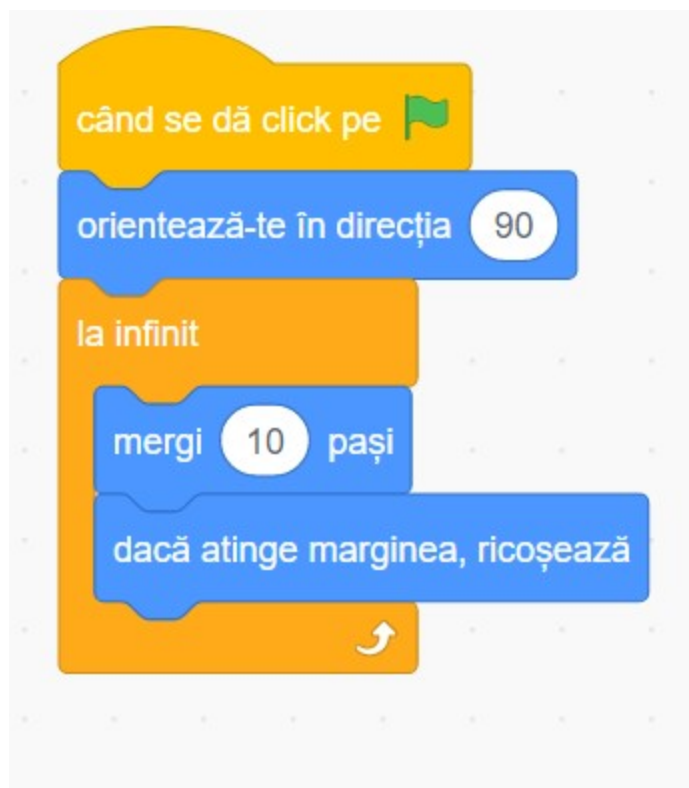
Introducere



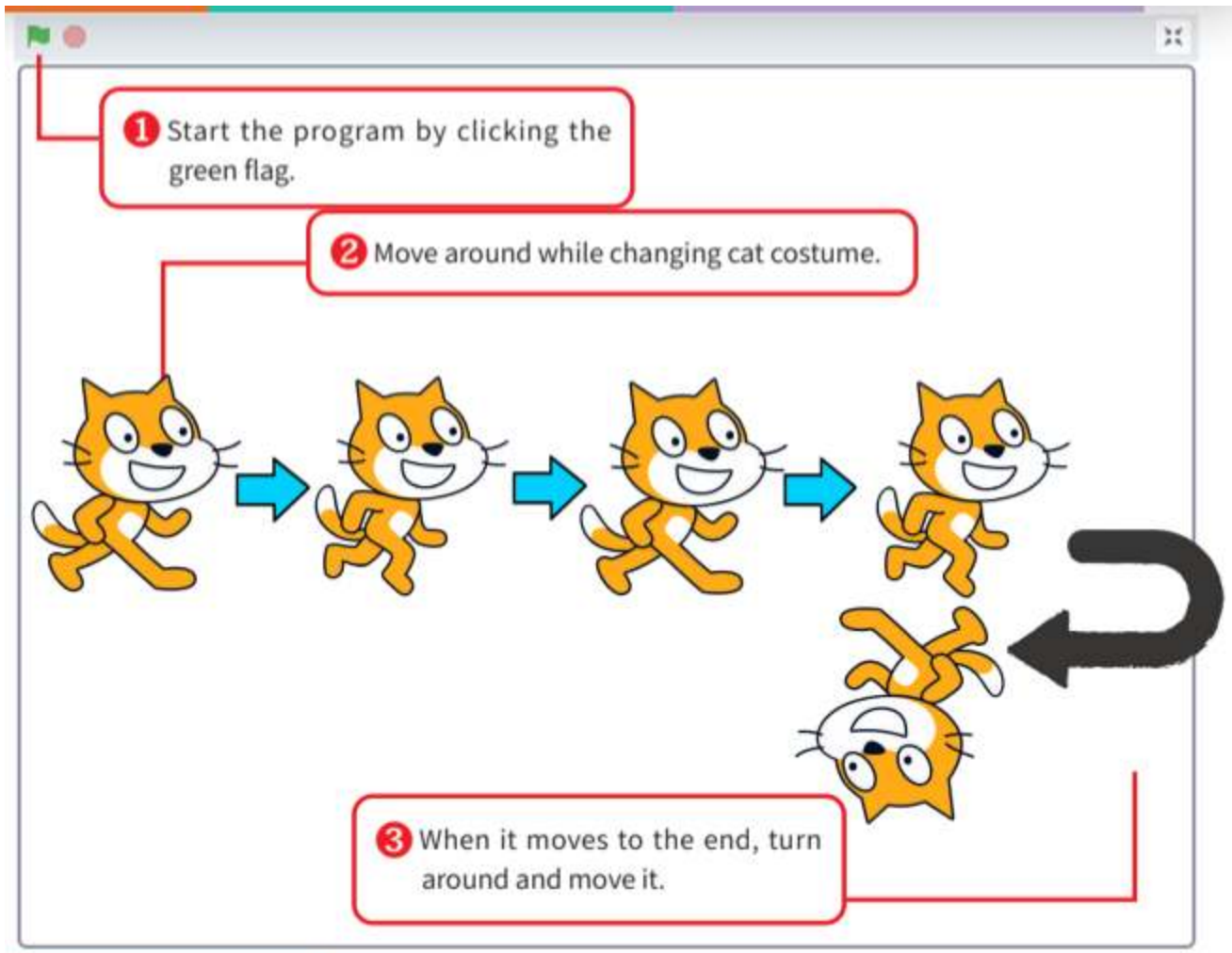
A8 Animatie Plimbare



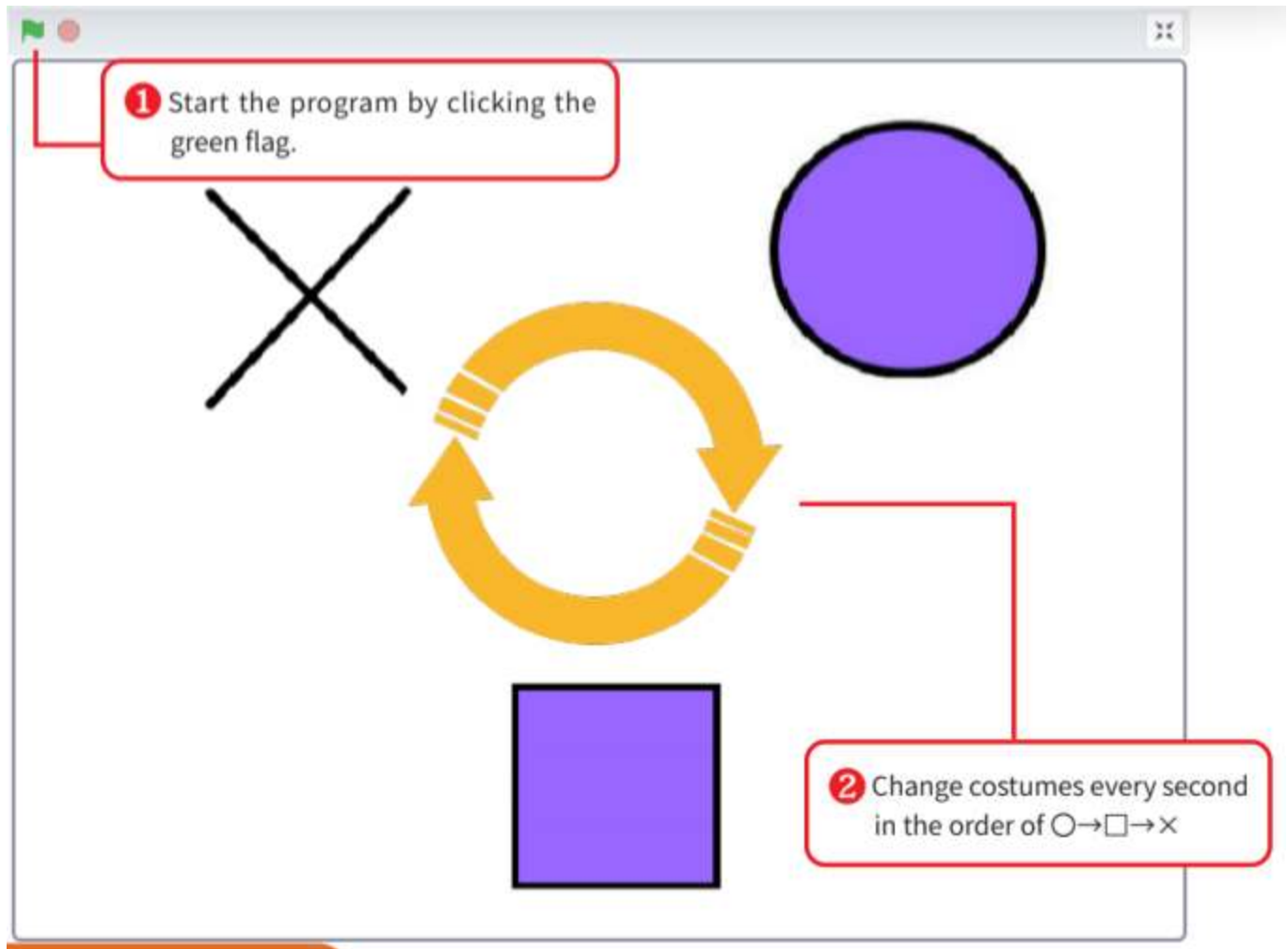
Introducere



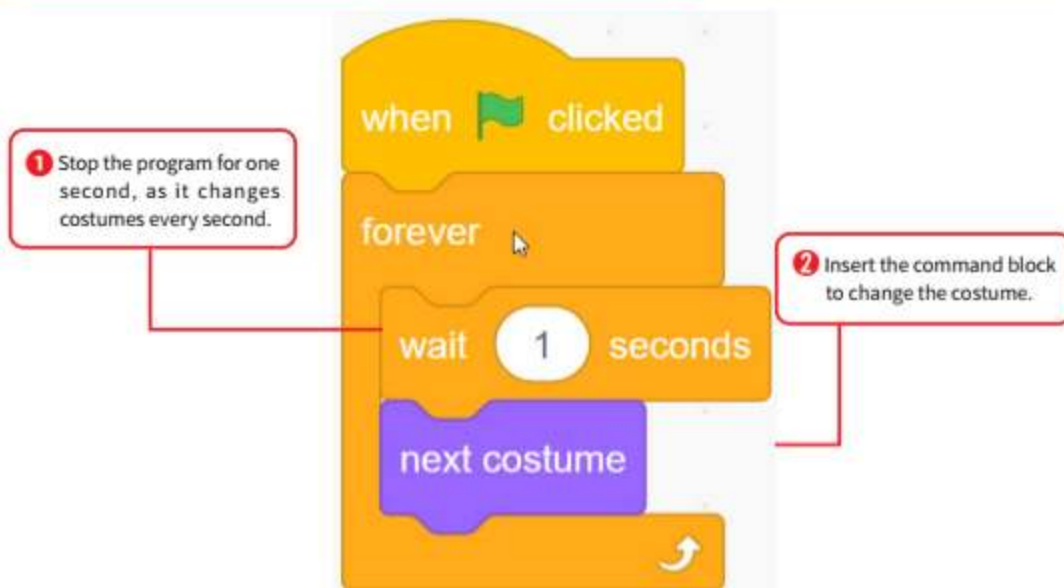
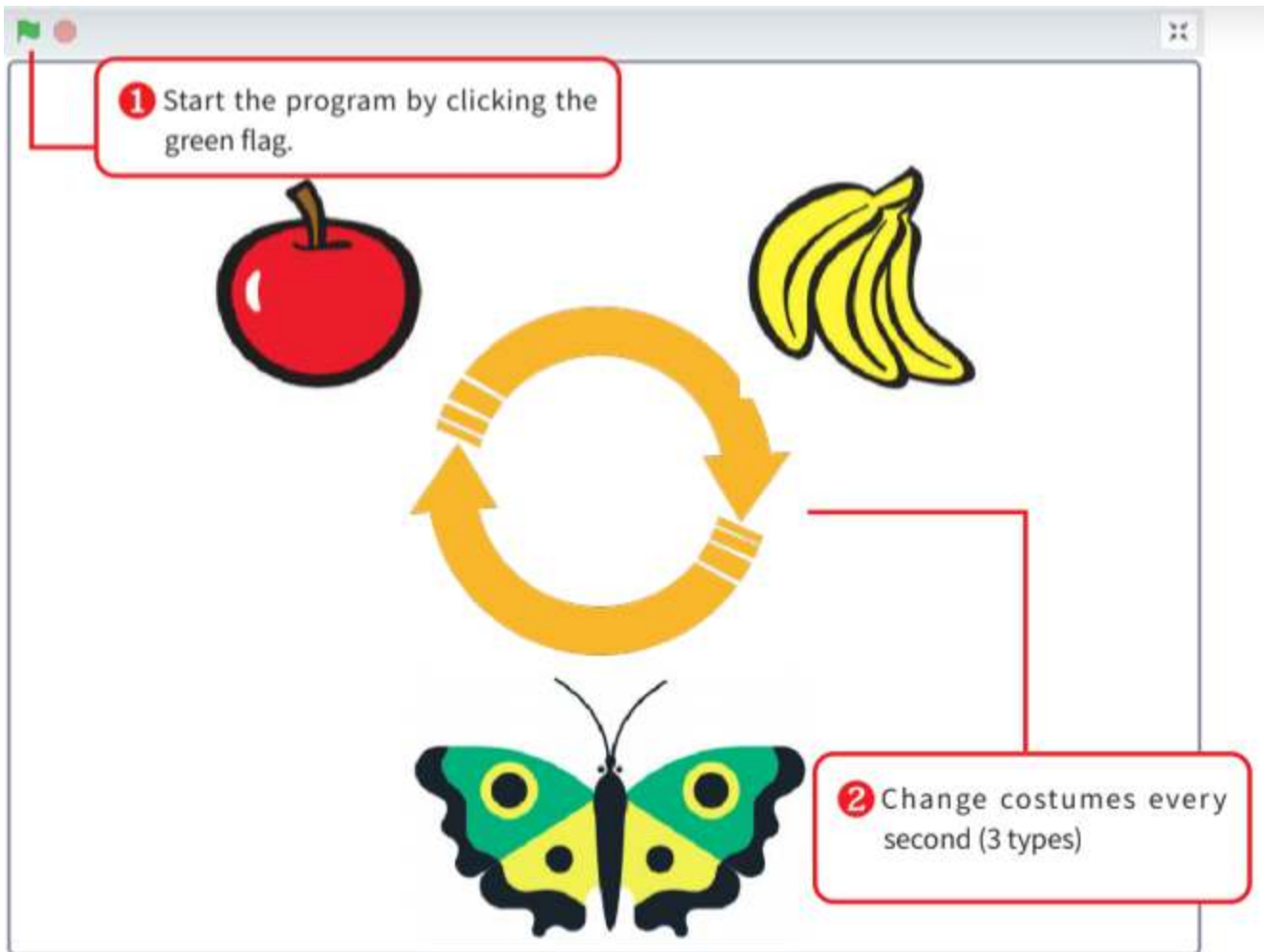
Introducere



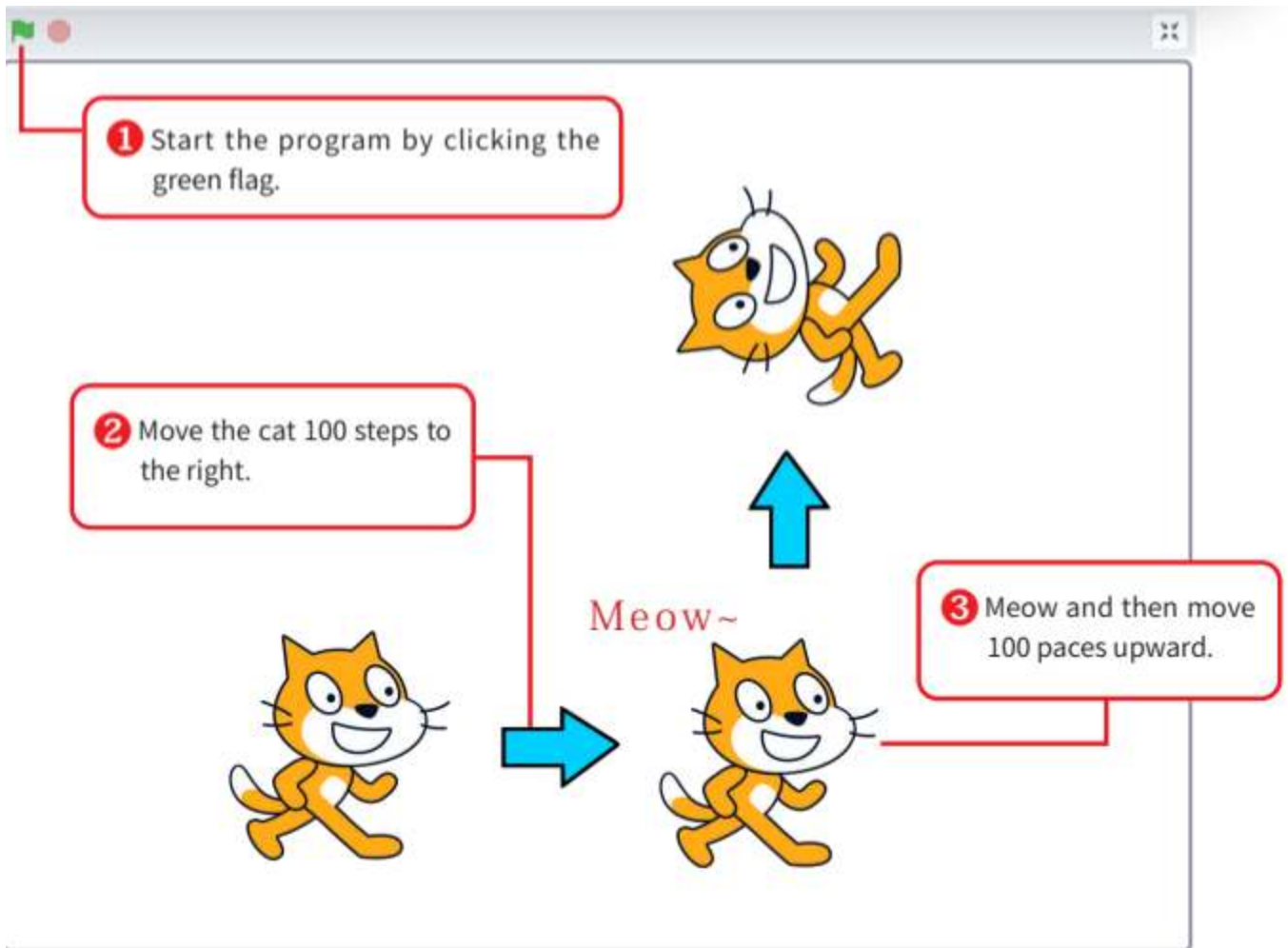
A10 Mai multe costume



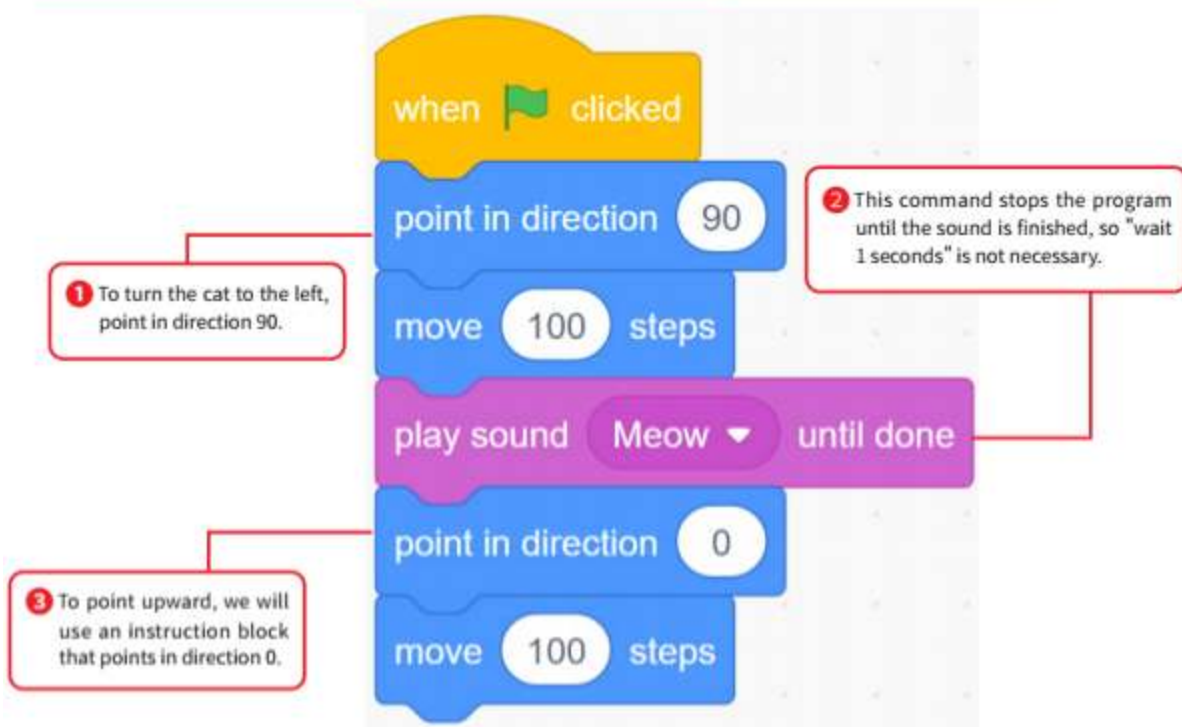
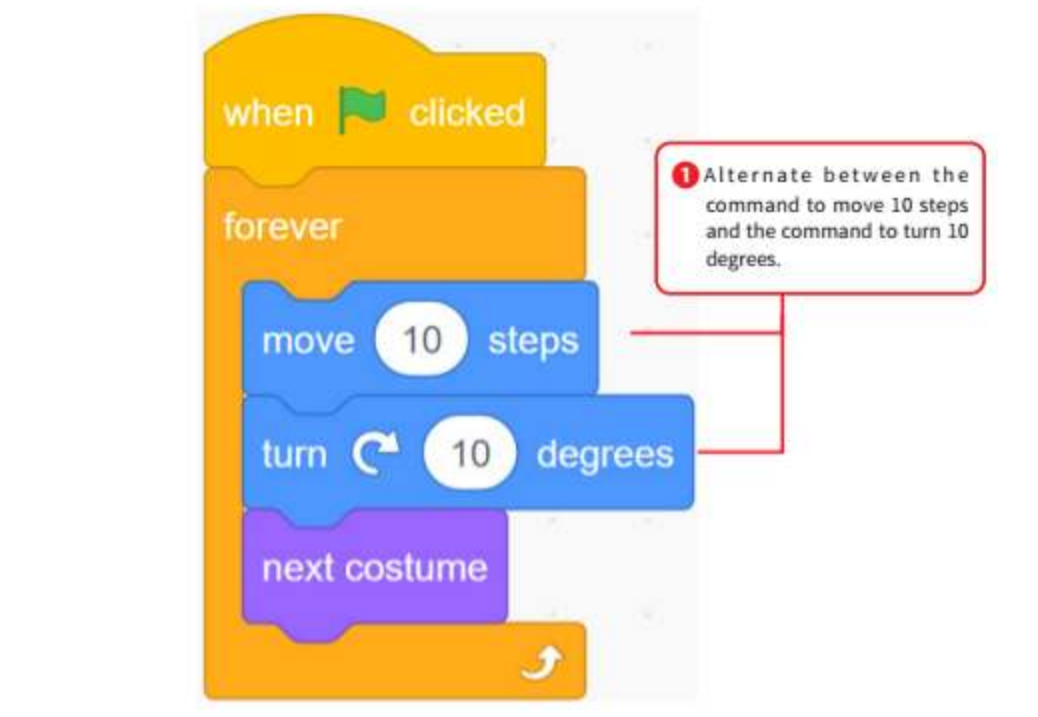
A11 Mai multe costume 2



A12 Mieunat si miscare



Introdúcere

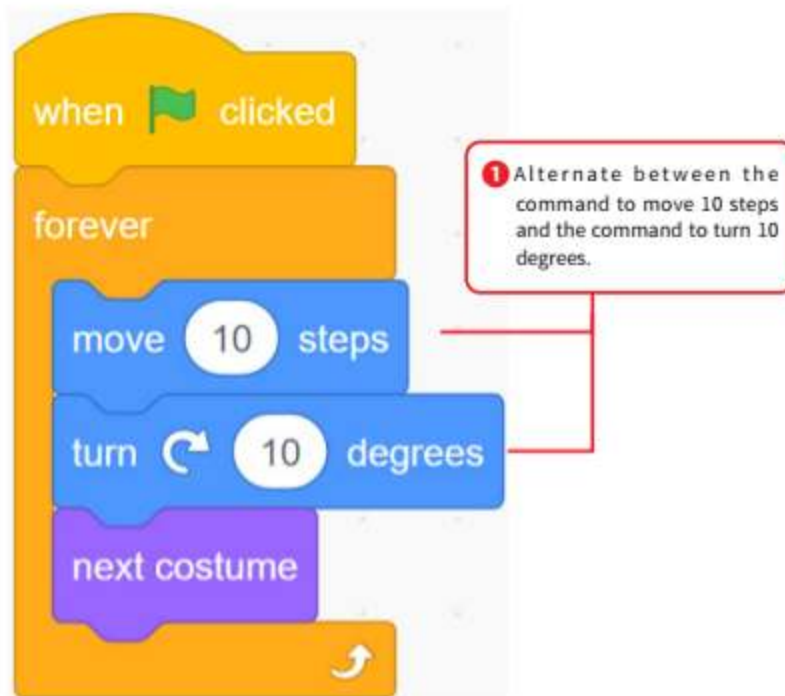
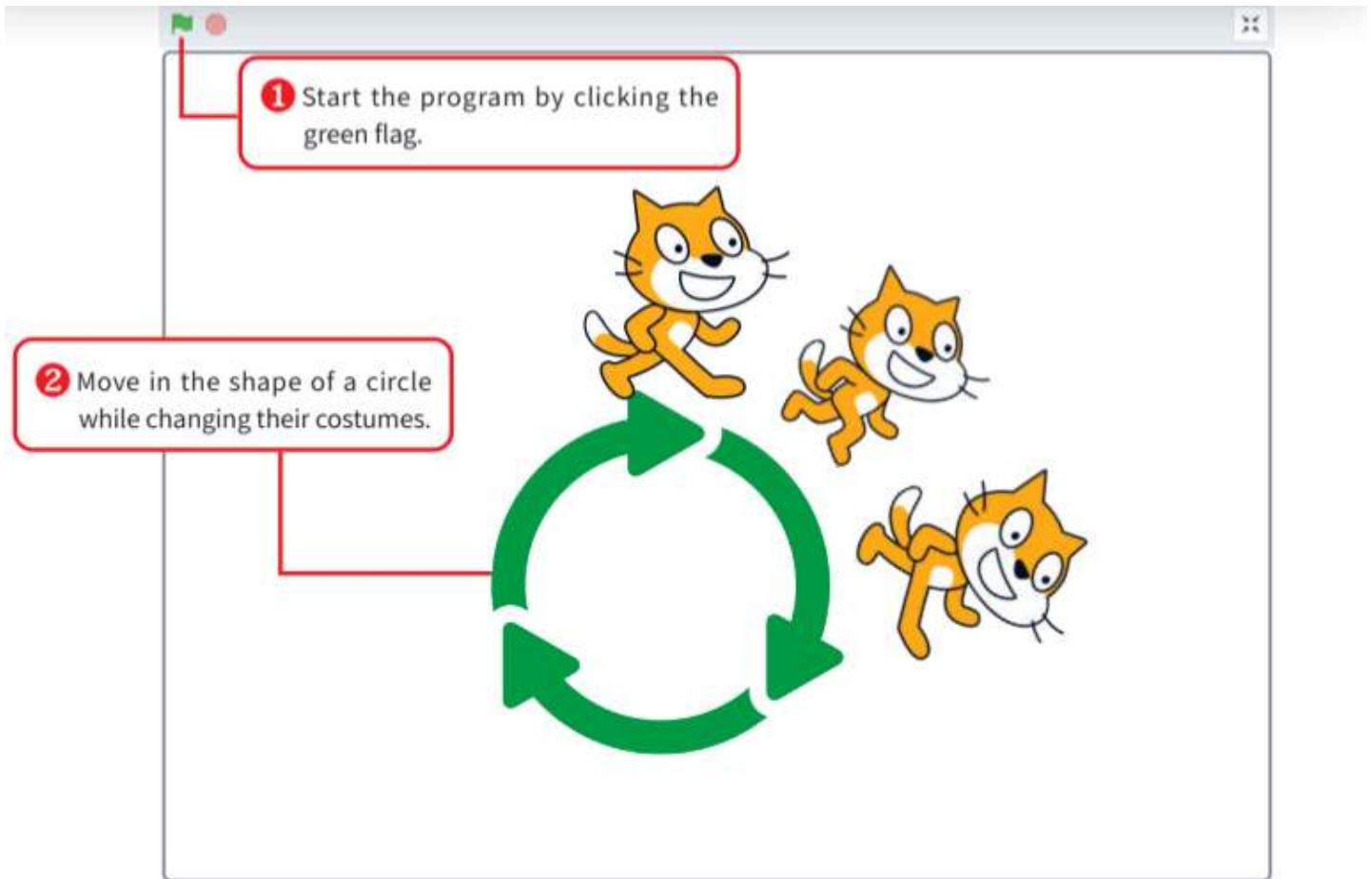


Caiet de Activitati Nr1-Scratch

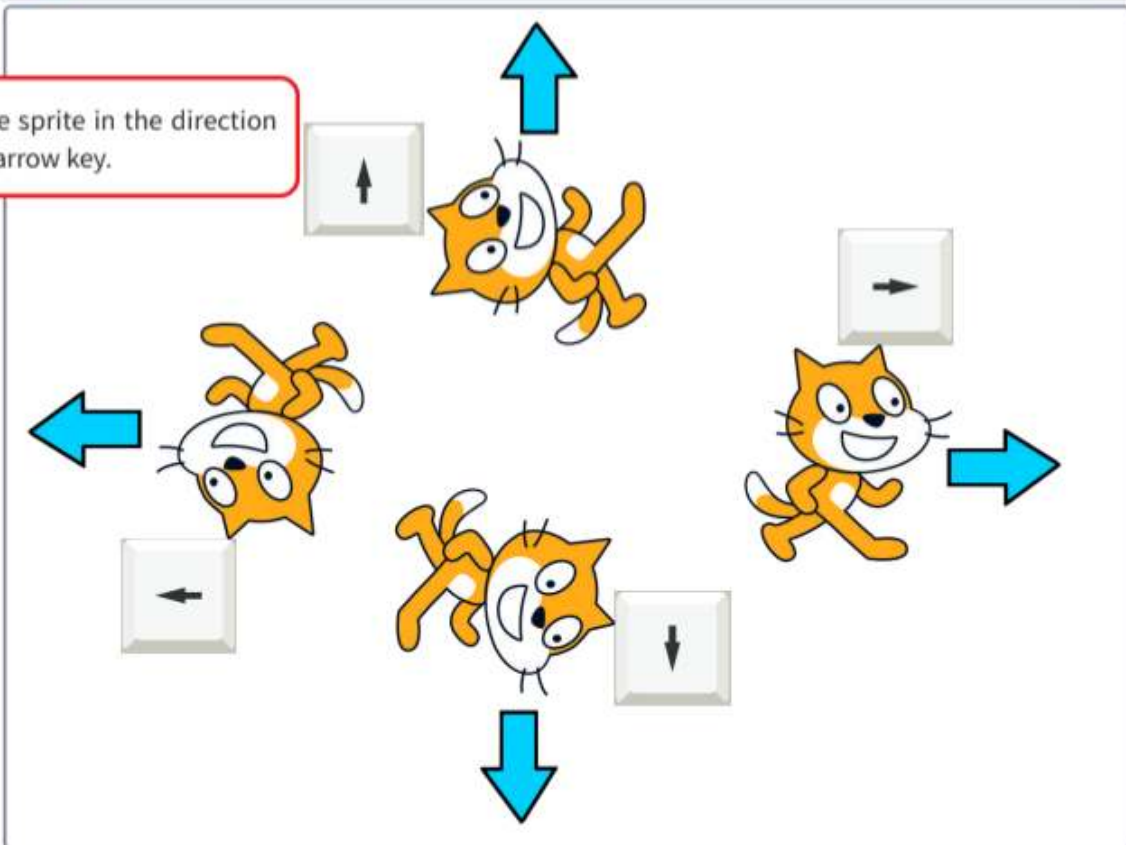
Level 2

A13 Miscare in forma de cerc

Introdurre



A14 Miscare cu sagetile



Move the sprite in the direction of each arrow key.

1 The program will start when the sprite is clicked.

when this sprite clicked

move 100 steps

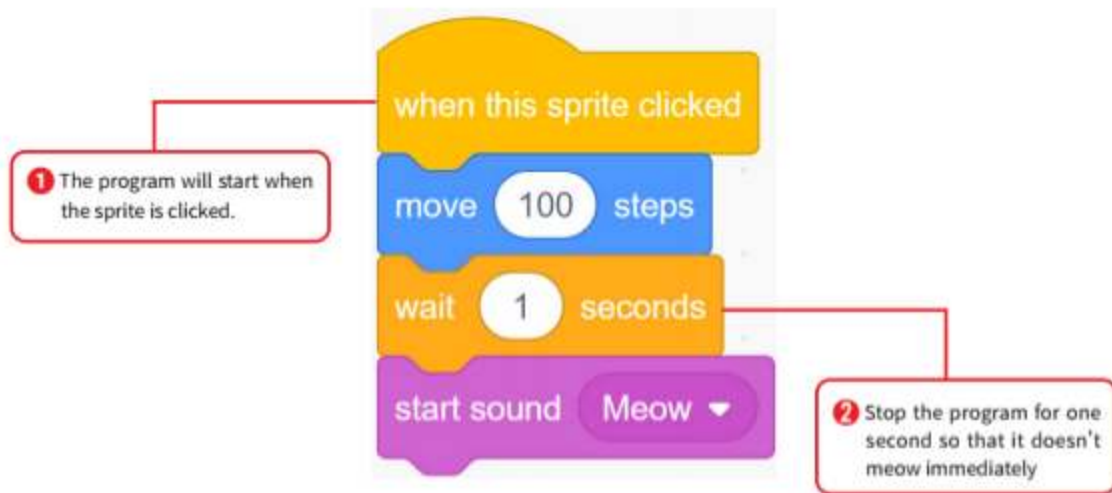
wait 1 seconds

start sound Meow

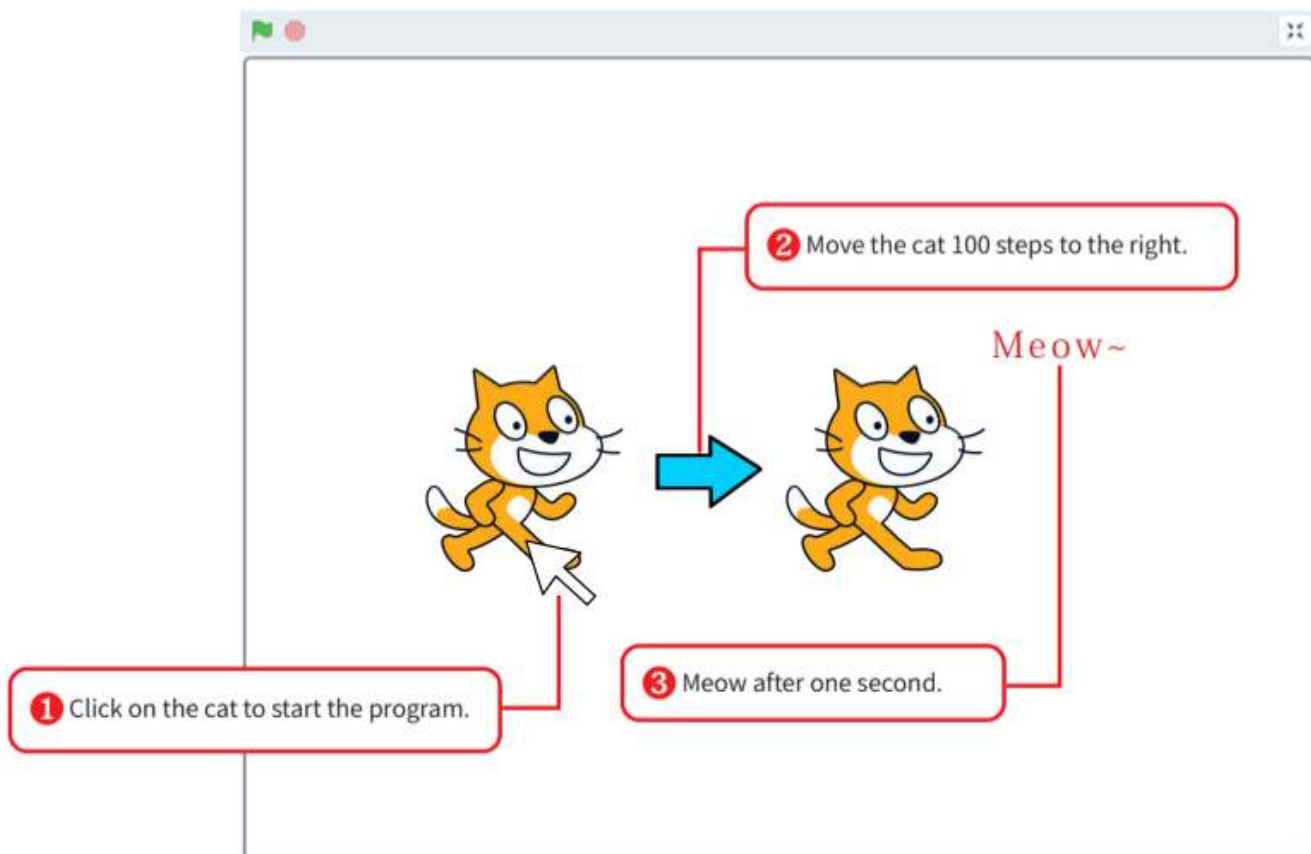
2 Stop the program for one second so that it doesn't meow immediately

The image shows a Scratch project window with a white stage. Four orange cat sprites are positioned around the center. Each cat is associated with a grey arrow key icon: an up arrow, a right arrow, a left arrow, and a down arrow. Large blue arrows point from each cat in the direction of its corresponding key. A red-bordered box in the top-left corner contains the text 'Move the sprite in the direction of each arrow key.' Below the stage, a script area shows a sequence of four blocks: a yellow 'when this sprite clicked' block, a blue 'move 100 steps' block, an orange 'wait 1 seconds' block, and a purple 'start sound Meow' block. A red-bordered box with the number '1' points to the 'when this sprite clicked' block, containing the text 'The program will start when the sprite is clicked.' Another red-bordered box with the number '2' points to the 'wait 1 seconds' block, containing the text 'Stop the program for one second so that it doesn't meow immediately'.

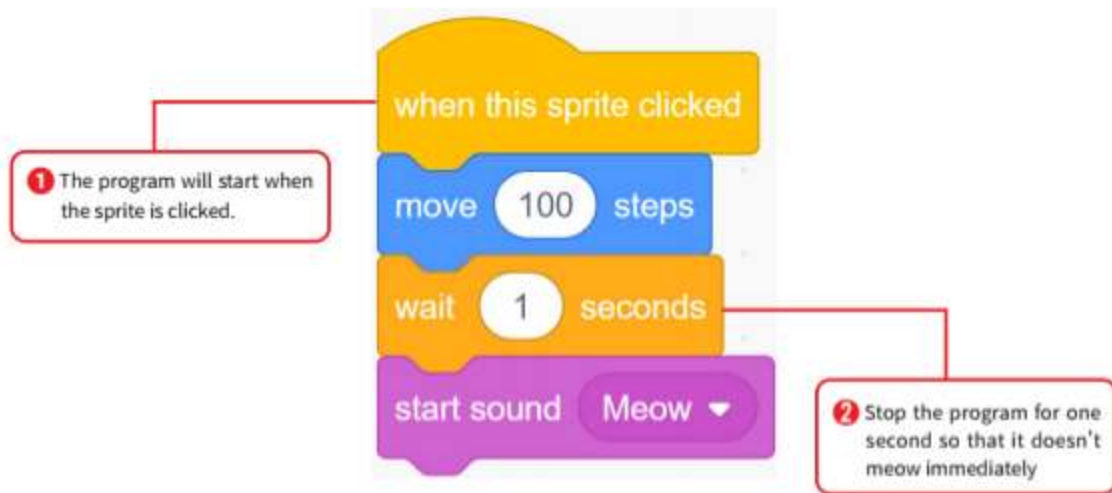
Introdurre



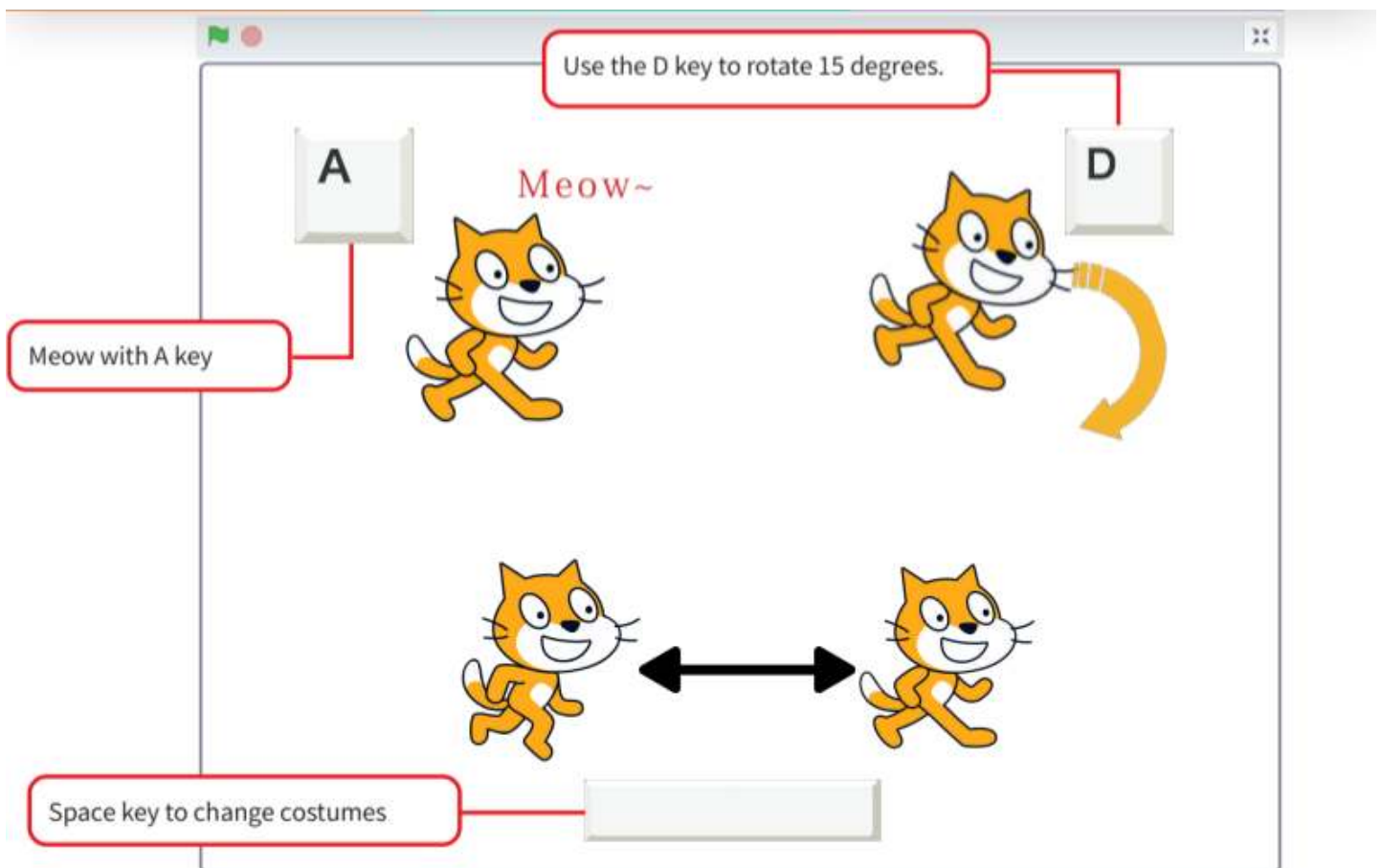
A15 Click Miscare sprite



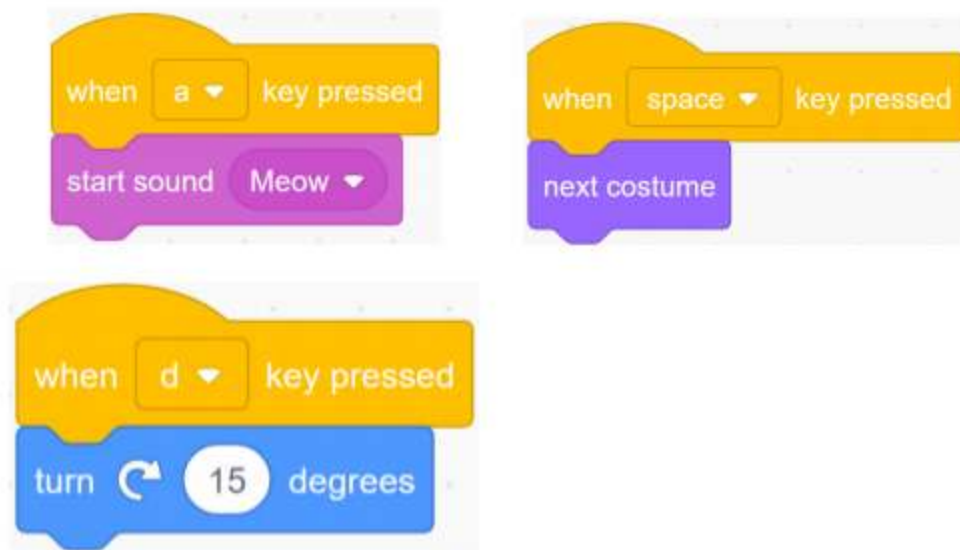
Introducere



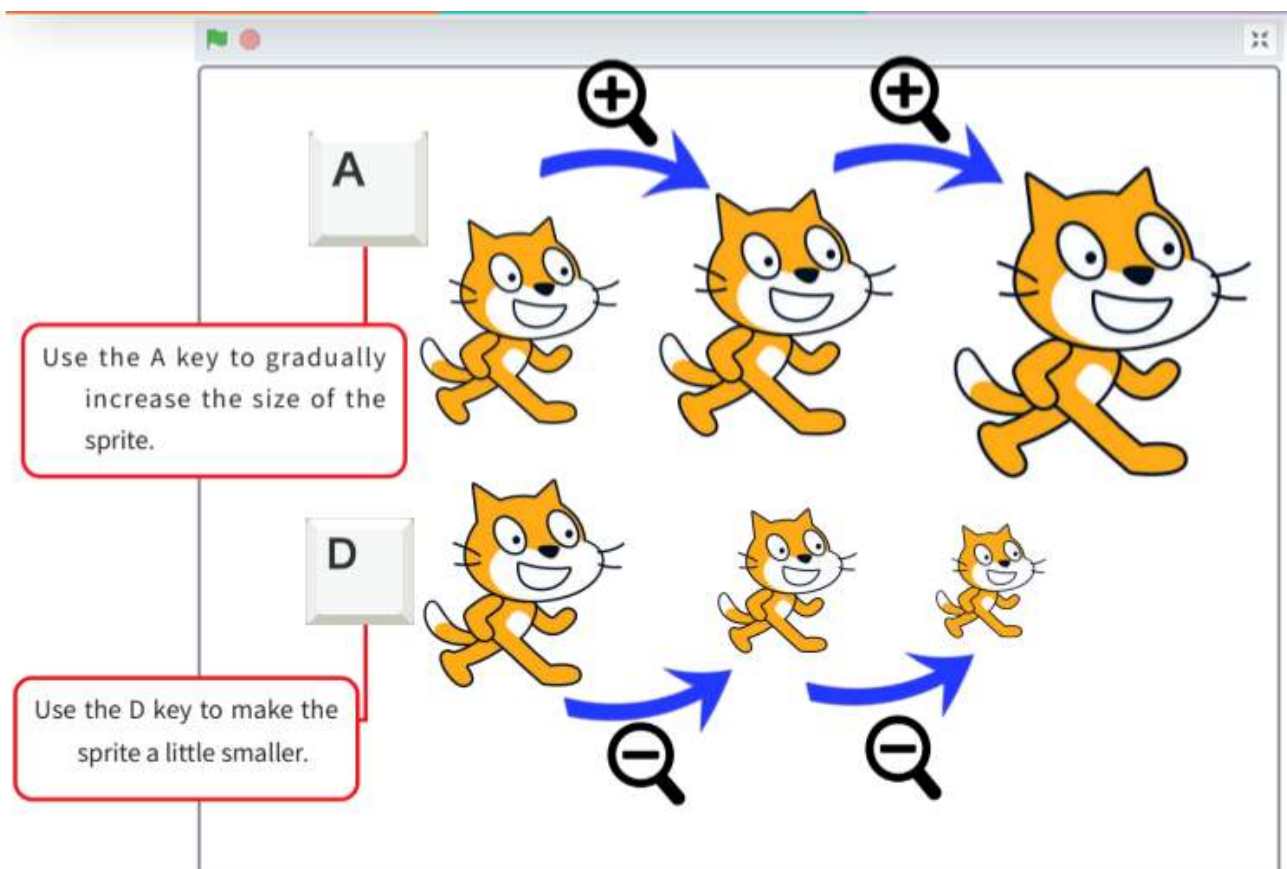
A16 Folosiți o varietate de taste (introducere).



Introducere



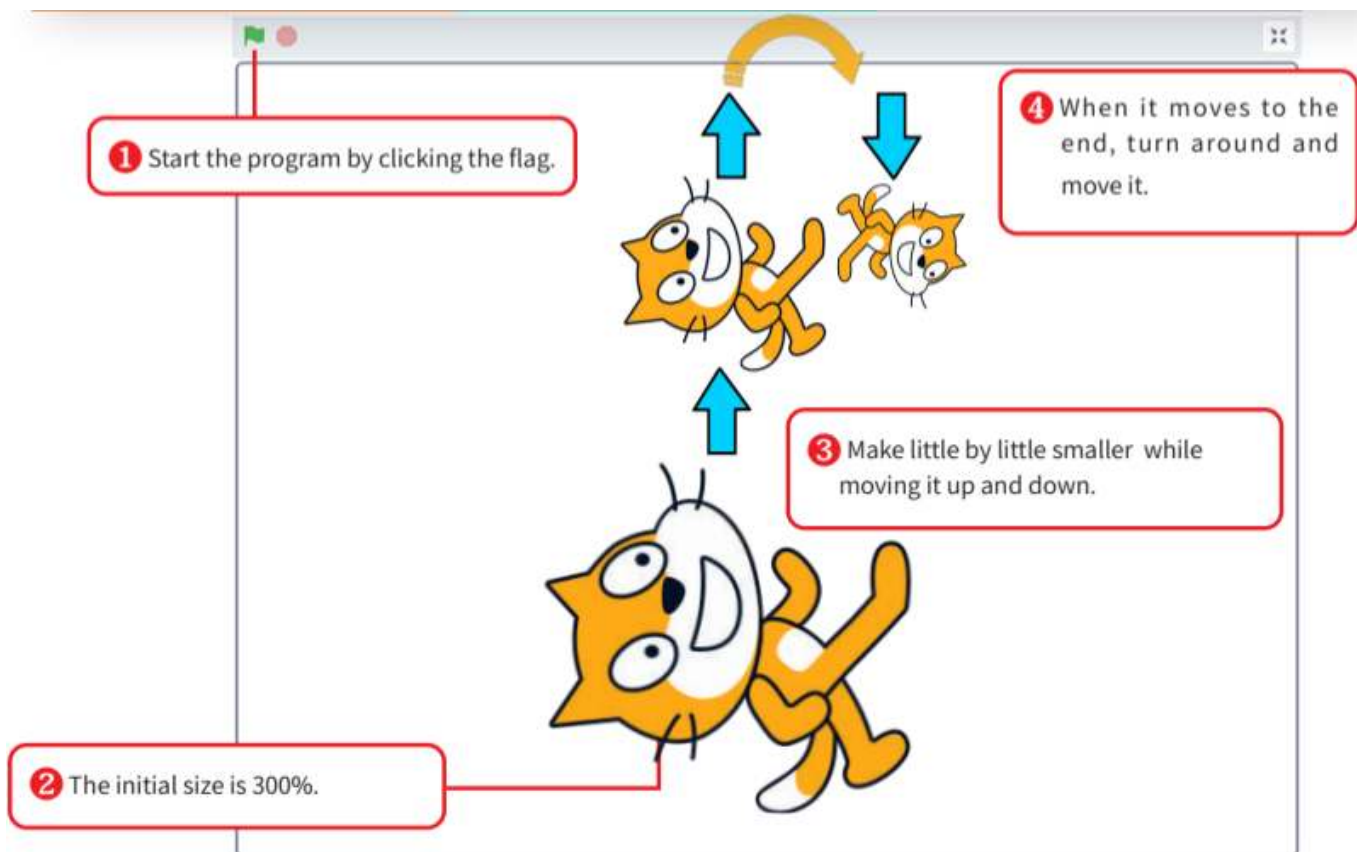
A17 Să schimbăm dimensiunea.



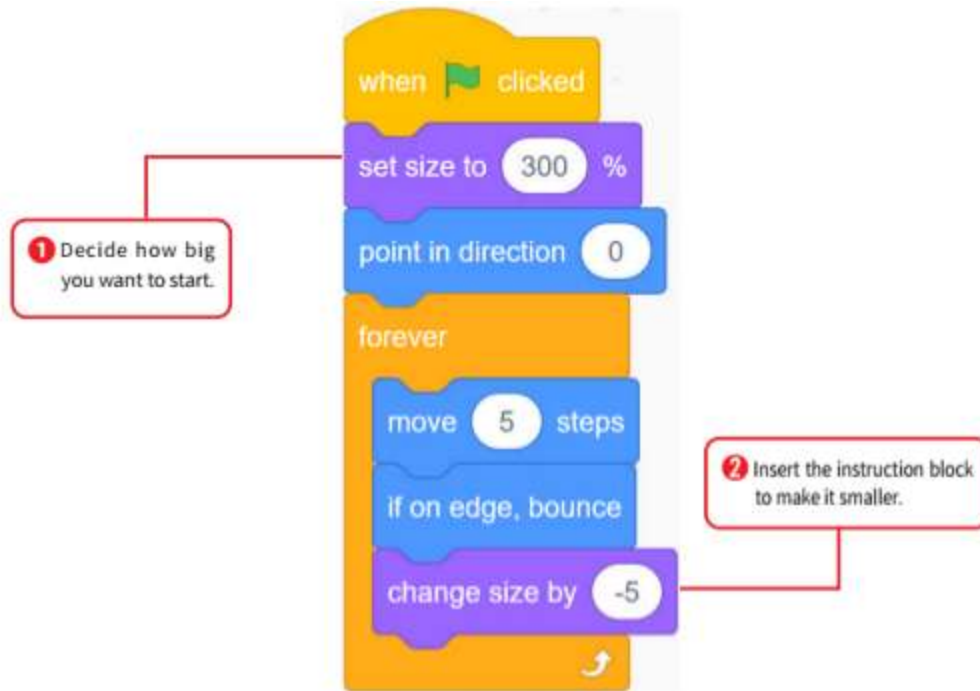
Introducere



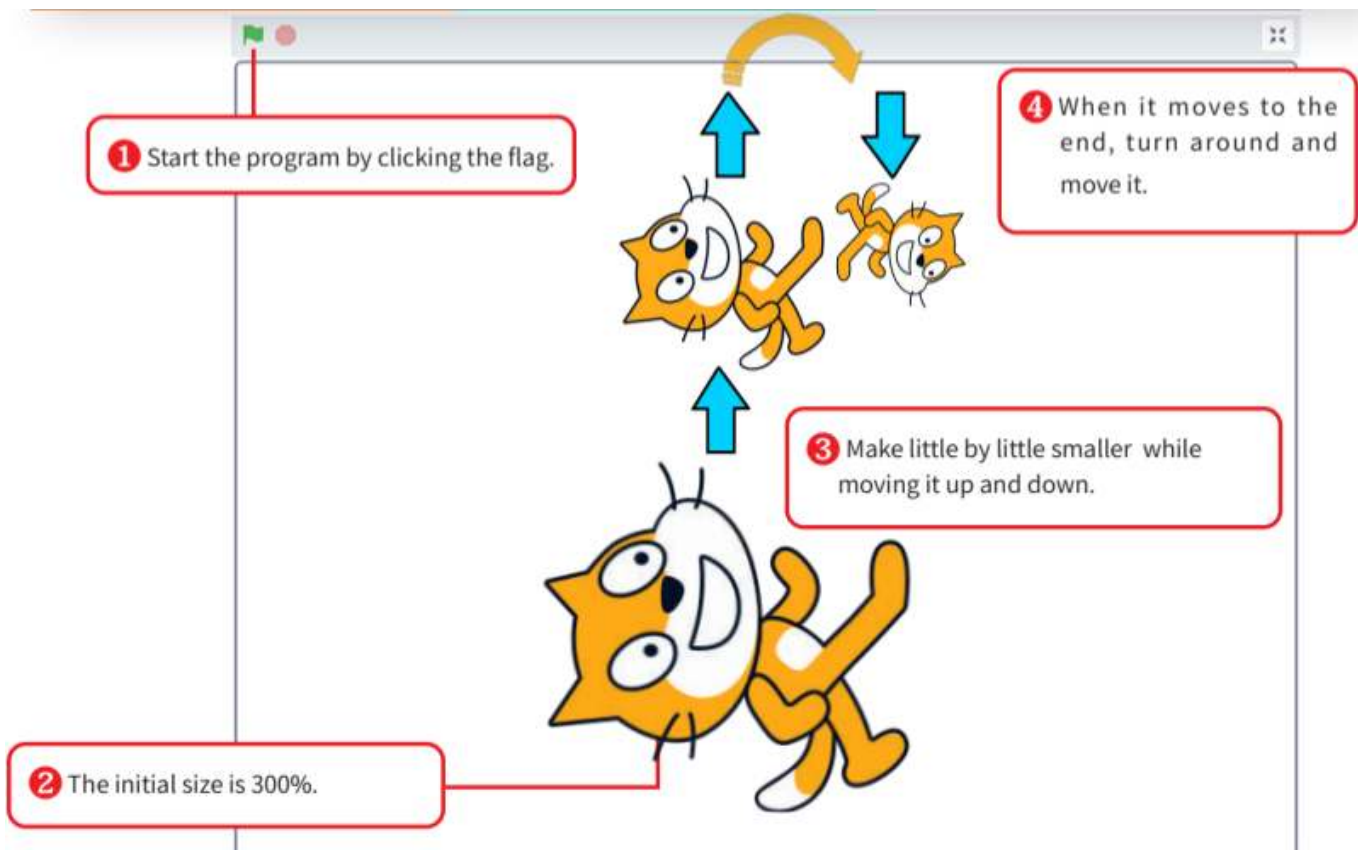
A18 Să-l facem mai mare mișcându-l.



Introducere

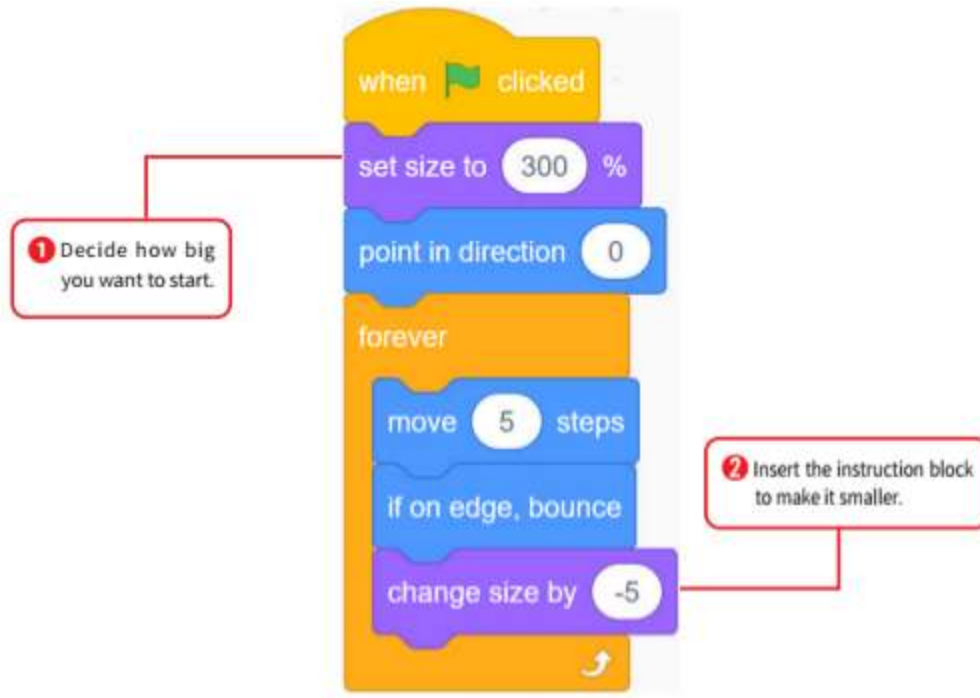


A19 Să-l micșorăm mutându-l.



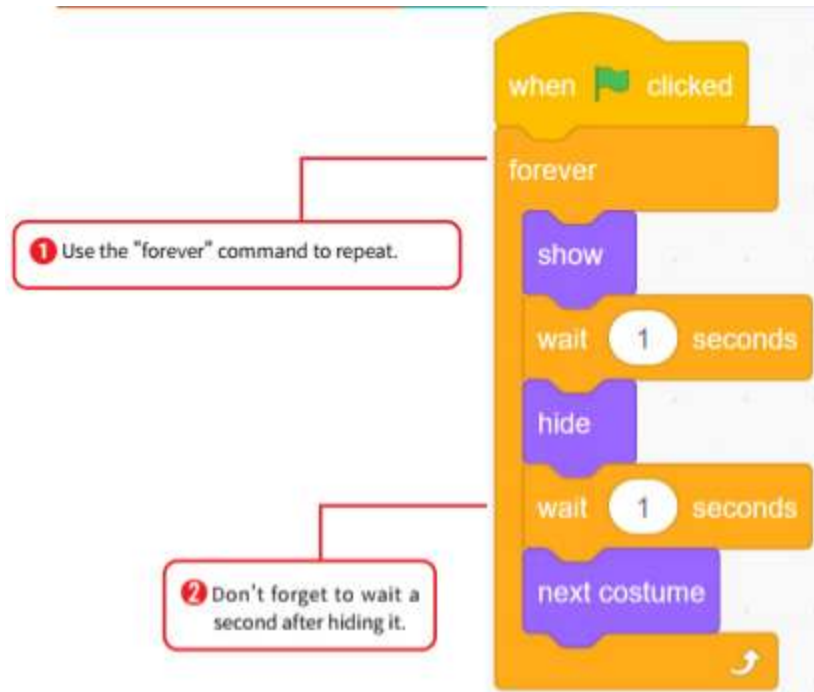
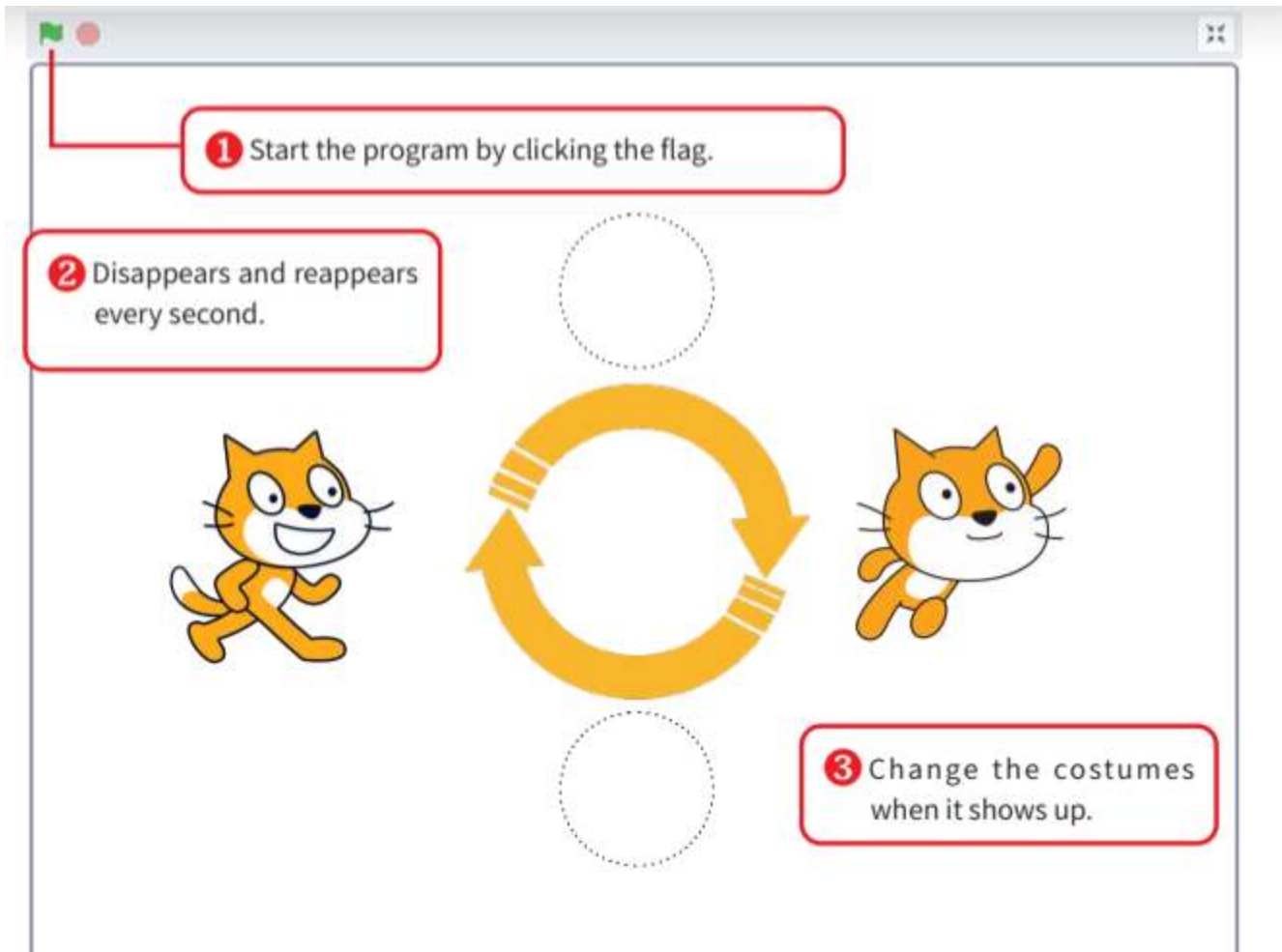
Introducere

Introducere

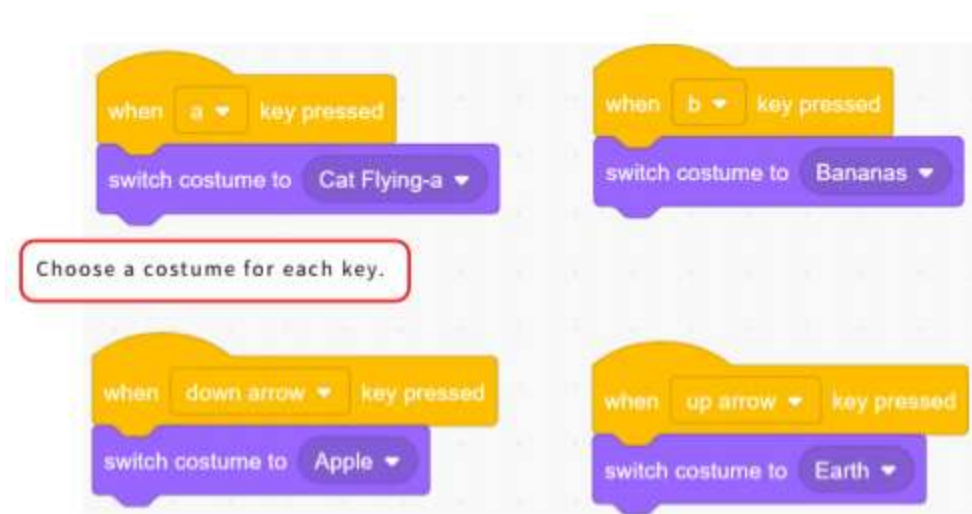
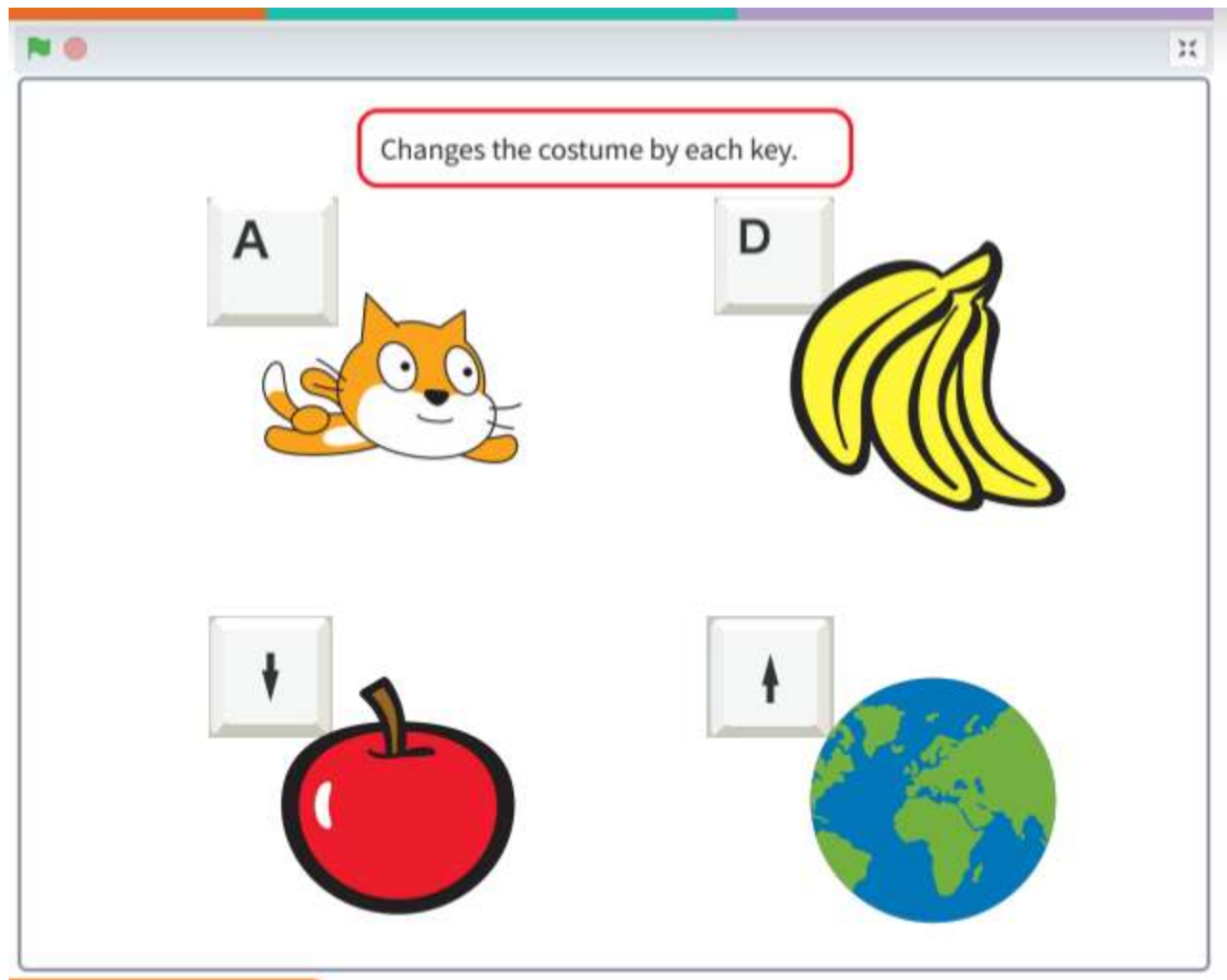


A20 Intermitent și schimbare costum.

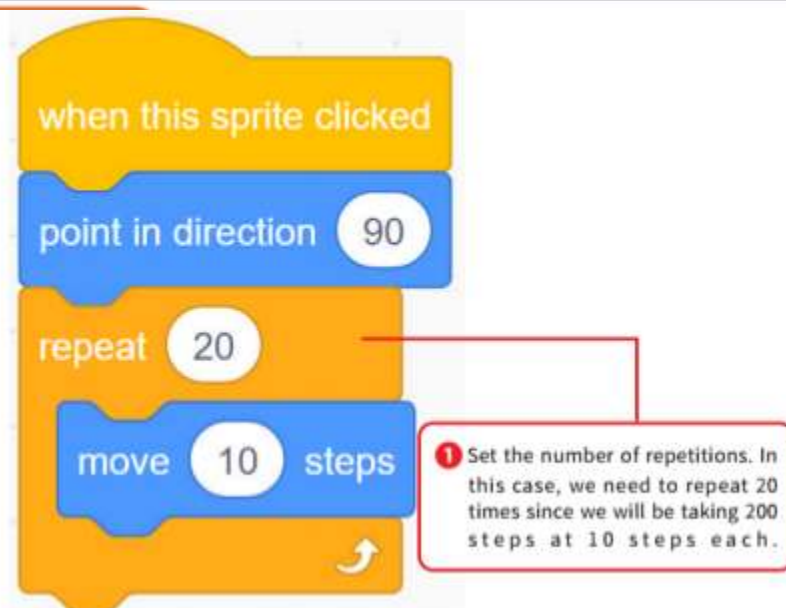
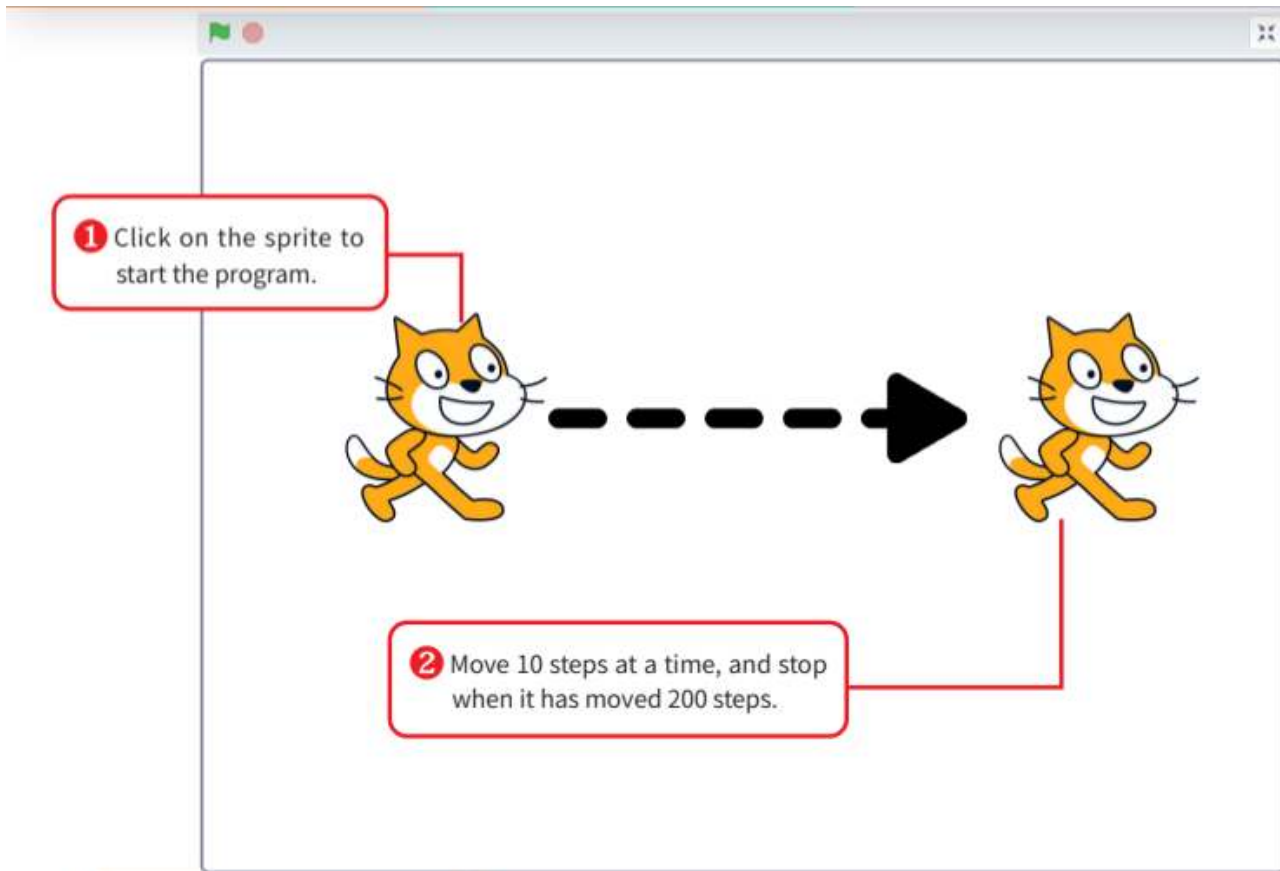
Introdurre



A21 Alegeți costumul direct.

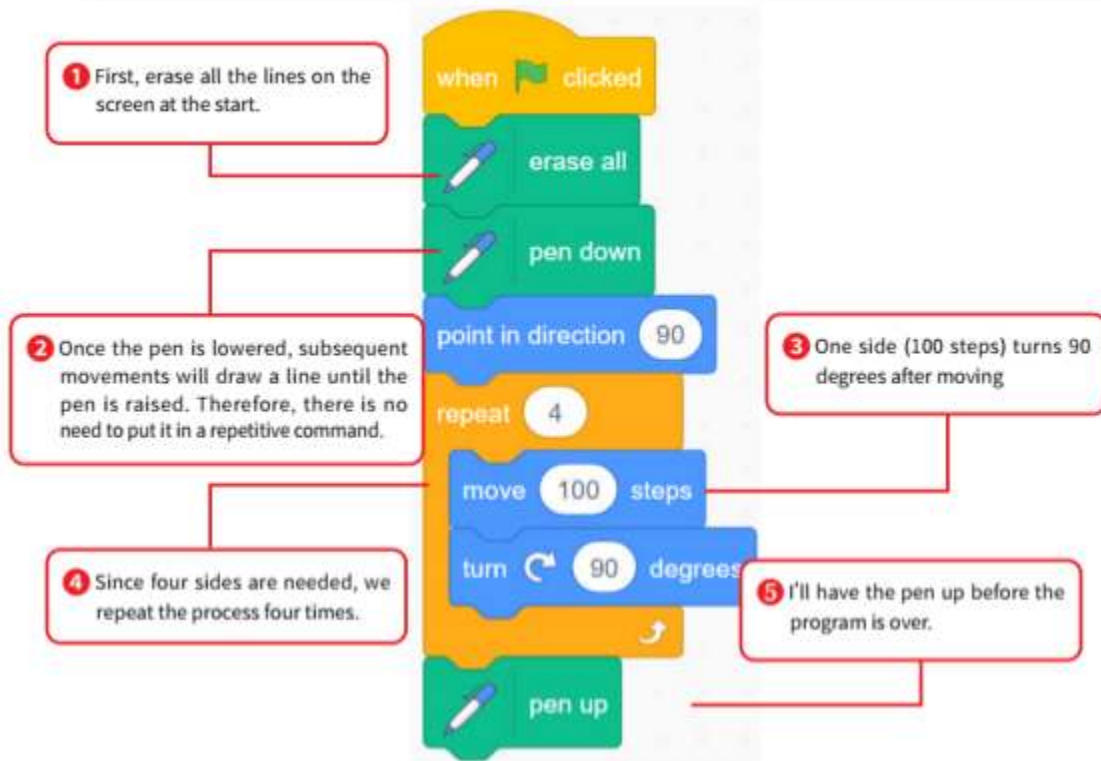
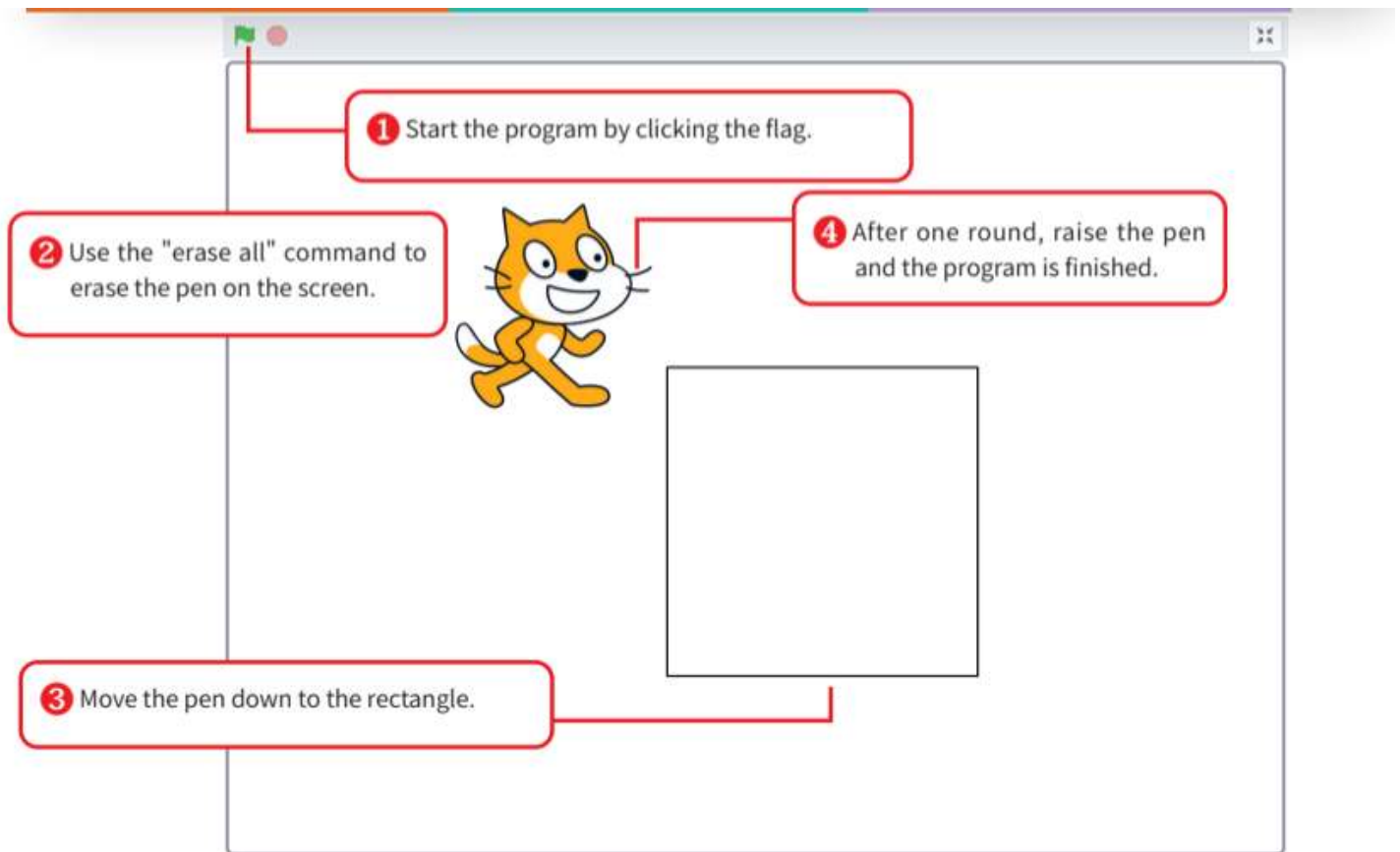


A22 Repetare cu un număr fix de ori.



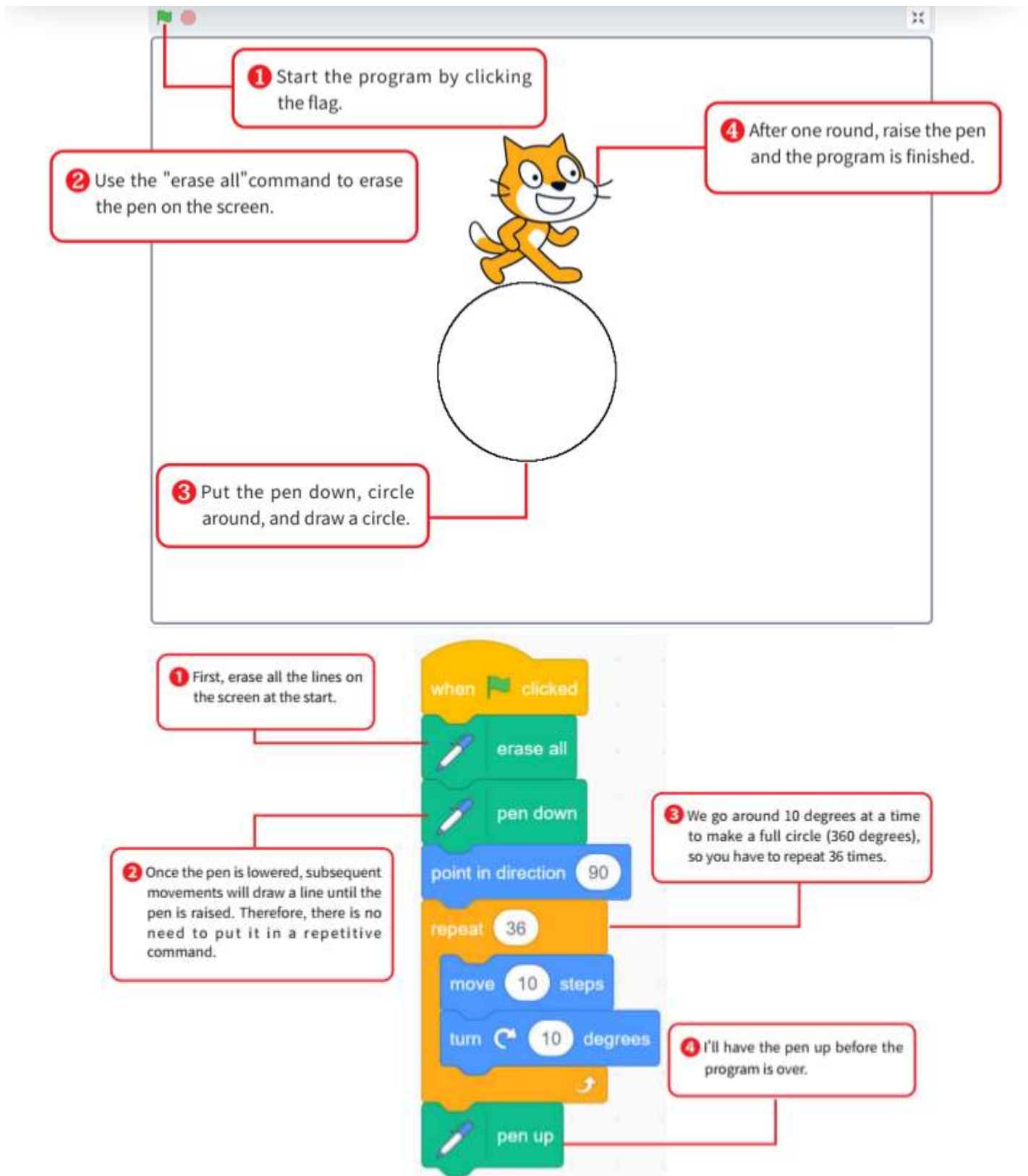
A23 Să desenăm un dreptunghi cu un stilou.

Introducere



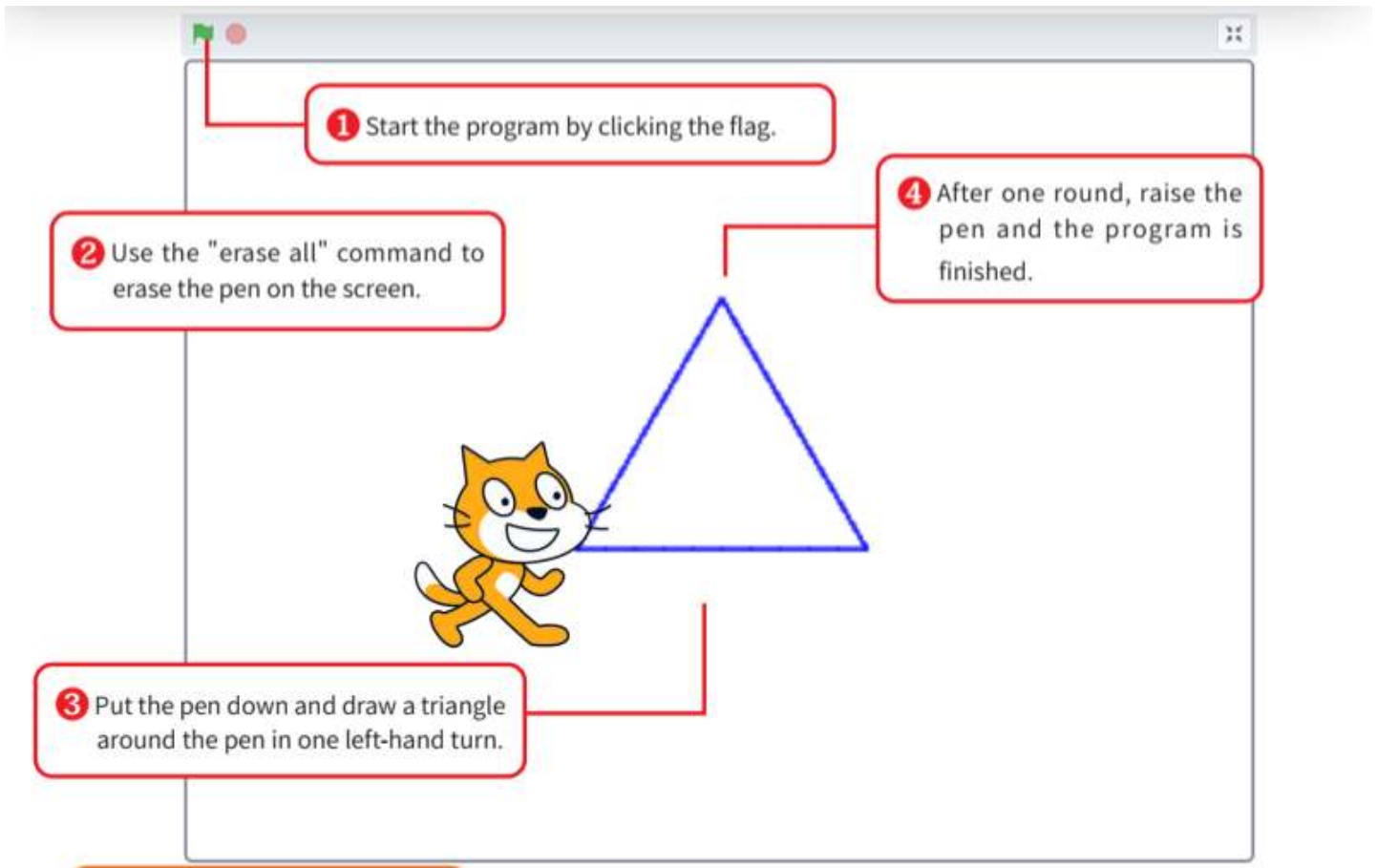
A24 Să desenăm un cerc cu un stilou.

Introducere

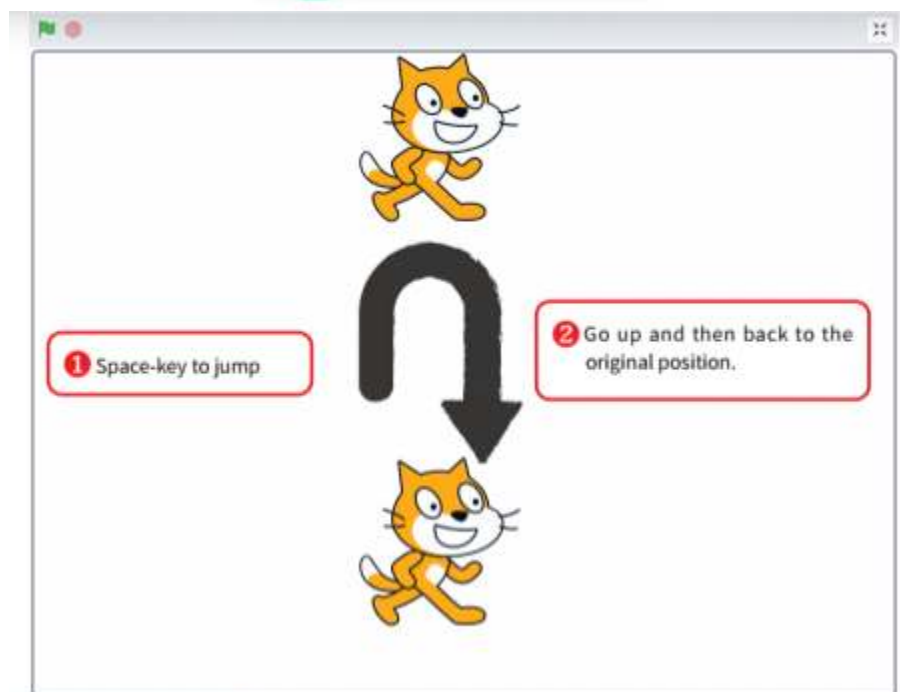
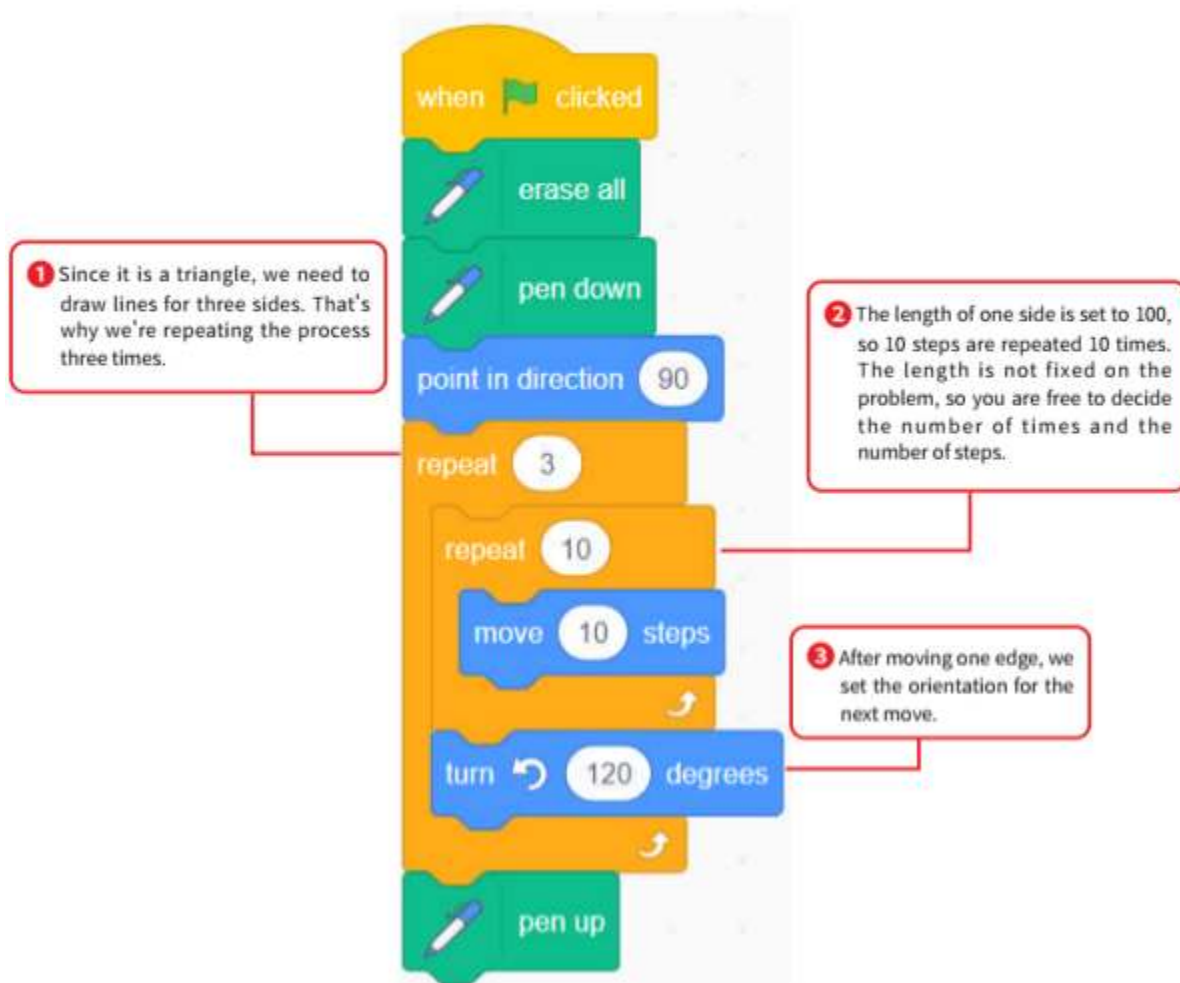


A25 Să desenăm un triunghi cu un stilou.

Introdurre



Introdurre



Introdurre

