LATVIJAS UNIVERSITĀTE

PEDAGOĢIJAS, PSIHOLOĢIJAS UN MĀKSLAS FAKULTĀTE
PIEAUGUŠO PEDAGOĢISKĀS IZGLĪTĪBAS CENTRS

DATORIKAS FAKULTĀTE

ALGORITMU MĀCĪŠANAS IESPĒJAS PAMATSKOLĀ ĀRPUSKLASES NODARBĪBĀS

DIPLOMDARBS

Autors: **Nora Meire**
Studenta apliecības Nr.: 11007
Darba vadītājs: lektore Iveta Gultniece

RĪGA 2013

ANOTĀCIJA

Diplomdarbā pētīta algoritmu mācīšanas nozīme un iespējas pamatskolā. Darba teorētiskajā daļā aplūkoti dažādu autoru viedokļi par algoritmu mācīšanas iespējām un nepieciešamību. Tika pētīts darbs ar algoritmiem, izmantojot dažādas programmēšanas vides, kā arī apkopoti skolēnu un vecāku atzinumi par strukturētās domāšanas atttīstīšanas nozīmi tālākājā izglītībā.

Darbā apskatīti vispārējās pamatizglītības standarti mācību priekšmetā informātika Latvijā, Krievijā un Anglijā. Darba pamatdaļā aprakstītas trīs vizuālās programmēšanas valodas algoritmiskās domāšanas veidošanai. Apkopoti eksperimentālā pētījuma un anketēšanas rezultāti.

Izstrādāts tematiskais plānojums un metodisks materiāls algoritmu mācīšanai ārpusstundu nodarbībās pamatskolā.

Darbs sastāv no trīs nodaļām un diviem pielikumiem.

Darba kopējais lappušu skaits – 75 lpp.

Izmantoto avotu skaits – 15.

ATSLĒGAS VĀRDI

Algoritms

Pamatskola

Programmēšana

Vizuālās programmēšanas valodas

ANNOTATION

In diploma work is investigated significance and possibilities of teaching of algorithm in primary school. In the theoretical part of the diploma work opinions of various authors about possibilities and necessity of teaching of algorithm are explored. Work with algorithm was investigated using different environments of programming and as well as opinions of students and their parents about structured thinking development importance in their further education were summarized.

In work are explored standards of general education in subject informatics in Latvia, Russia and England. In the main part of the work three visual programming languages for forming of algorithmic thinking are described. Results of experimental research and questionnaires are summarized.

Thematic planning and methodical material for teaching the algorithms in extracurricular lessons in primary school is elaborated.

The work consists of three parts and two appendixes.

Total number of pages in the work is: 75 pages.

The number of sources: 15

KEYWORDS

Algorithm

Elementary school

Programming

Visual programming languages

SATURS

IEVADS

Mūsdienu skolotājs māca bērnus, gatavo nākotnes profesijām, kuras vēl nav radītas. Iespējams, ka jau tuvākajā nākotnē visa cilvēka darbība balstīsies uz datoru un to programmatūras darbu[13]. Informācijas un komunikācijas tehnoloģijas nozare ir piedzīvojusi strauju izaugsmi, datorika ir kļuvusi par daudzu zinātņu un nozaru neatņemamu sastāvdaļu. Nepārtraukti notiek izmaiņas informācijas tehnoloģiju industrijas jomā. Pieaug organizāciju atkarība no informācijas tehnoloģijas resursu izmantošanas, līdz ar to pieaug informācijas tehnoloģiju loma valsts ekonomisko un sociālo procesu vadībā.

Latvijā ir liels pieprasījums pēc augstas kvalifikācijas sistēmas analītiķiem, datoru tīklu speciālistiem, augstas kvalifikācijas programmētājiem, datorizētu sistēmu administratoriem. Šis pieprasījums ir ievērojami lielāks, nekā augstskolas pašreiz spēj sagatavot. Augstskolu uzdevums ir sagatavot augsti kvalificētus speciālistus, kuri spēj veikt sarežģītu informācijas sistēmu projektēšanu, izstrādāt sarežģītas lietojumprogrammas, radīt konkurētspējīgu informācijas tehnoloģiju produkciju un patstāvīgi apgūt jaunas tehnoloģijas un pilnveidoties. Latvijā lielāku vērību vajadzētu pievērst informātikai kā zinātnes nozarei. Tās attīstība varētu nodrošināt kvalificētu speciālistu sagatavošanu.

Vidējās mācību iestādes gatavo studijām augstskolā. Programmēšanas pamati kā mācību priekšmets tiek iekļauts dabaszinību, tehnoloģiju un matemātikas izglītības programmas mācību plānā un tiek mācīti vienu vai divus gadus, turklāt var būt kā izvēles priekšmets. Sākt mācīties programmēt augstskolā vai vidusskolā ir par vēlu. Par to liecina statistika – no imatrikulētiem studentiem IKT jomā absolvē aptuveni tikai puse. Lielākā daļa pārtrauc studijas jau pirmajos semestros. Ir liela atšķirība starp vidusskolu un augstskolu gan nepieciešamajās zināšanās, gan mācību procesā. Situācija varētu uzlaboties, ja programmēšanas elementi tiktu iekļauti mācību procesā agrāk – pamatskolā vai jau sākumskolā. Tas sekmētu loģiskās domāšanas attīstīšanu, strukturētās domāšanas attīstību un pavērtu plašākas iespējas iekļauties visas pasaules darba tirgū.

Darba hipotēze: algoritmu apguve ārpusstundu nodarbībās, izmantojot vizuālās programmēšanas programmas, ir vienkārša un saprotama.

Darba mērķis: izpētīt algoritmu mācīšanas iespējas pamatskolā.

Darba uzdevumi:

1) analizēt literatūru par algoritmu nozīmi strukturētas domāšanas attīstīšanā;
2) veikt pētījumu algoritmu mācīšanu pamatskolā, apkopot pētījuma materiālu;

3) izstrādāt tematisko plānojumu un praktisku uzdevumu krājumu par algoritmu mācīšanas iespējām pamatskolā.

Pētījuma objekts: algoritms.

Pētījuma priekšmets: algoritmu apguve ārpusstundas nodarbībās.

Pētniecības metodes:

1. Teorētiskās:

1.1.Zinātniskās literatūras analīze.

2. Empīriskās:

2.1.Pedagoģiskais vērojums.

2.2.Pedagoģiskais eksperiments.

2.3.Aptauja.

Pētījuma bāze: X vidusskolas pamatskolas klases.

Darbam ir trīs daļas.

Pirmajā daļā ir analizēts algoritma jēdziens mācību literatūrā.

Otrajā daļā ir aplūkoti mācību priekšmeta „Informātika" standarti Latvijā, Krievijā un Anglijā, algoritmu mācīšanai atvēlētais stundu skaits.

Trešajā daļā ir izstrādāts tematiskais plānojums un praktisko darbu komplekts algoritmu pasniegšanai pamatskolā ārpusklases nodarbībās.

1. AGLORITMU APGUVE PAMATSKOLĀ

Algoritms ir viens no programmēšanas un visas datoru zinātnes centrālajiem jēdzieniem.

Algoritms ir matemātisks priekšraksts uzdevuma risināšanai. Precīzāk runājot, tā ir noteikta darbību virkne, kuras izpildot, tiek iegūts vajadzīgais rezultāts[7].

Matemātikā un datorzinātnē algoritms ir process kā soli pa solim veikt aprēķinus. Algoritmus izmanto kalkulatoros, datu apstrādē un automātiskā spriešanā. Precīzāk, algoritms ir efektīva metode, kura ir izteikta ar galīgu skaitu labi nodefinētām instrukcijām, lai aprēķinātu funkciju.

Vārds algoritms ir nācis no 9. gadsimta Persiešu musulmaņu matemātiķa Abu Abdullah Muhammad ibn Musa Al-Khwarizmi. Vārds algoritms sākotnēji attiecās tikai uz noteikumiem pēc kuriem veica aritmētiskas darbības ar Hindu-Arābu cipariem, bet ar Eiropiešu latīņu tulkojumu no Al-Khwarizmi vārda pārtapa algoritmā aptuveni 18. gadsimtā. Algoritma izmantošana apzīmē visas procedūras ar kurām risina problēmas vai paveic uzdevumus.

Dažādos avotos sastopamas mazliet atšķirīgas algoritma definīcijas, Šeit minētas dažas no tām:

- algoritms ir instrukciju virkne, kas nosaka noteiktas problēmas risināšanas veidu;
- algoritms ir precīzu, viennozīmīgu instrukciju virkne, kuru izpildes rezultātā tiek atrisināts noteikts uzdevums;
- algoritms ir stingra, precīza un galīga noteikumu sistēma, kas nosaka darbību virkni izpildīšanai uz noteiktiem objektiem un pēc galīgā soļu skaita sasniedz uzstādīto mērķi;
- algoritms ir sakārtots viennozīmīgu un izpildāmu etapu kopums, kas apraksta noteiktu pabeigtu procesu;
- algoritms ir mērķtiecīgs darbību izpildes priekšraksts, ar kuru no dotajiem sākumdatiem ar galīgu elementāru darbību jeb soļu skaitu iegūst rezultātu.

Algoritmiskā valoda ir apzīmējumu un likumu sistēma algoritmu vienveidīgam un precīzam pierakstam uz izpildei. Algoritmiskai valodai tāpat kā visām valodām ir sava vārdnīca, kuras pamatā ir vārdi ar kuriem pierakstīt komandas. Lai norādītu algoritma sākumu un beigas tiek lietoti komandas palīgvārdi: *sākums* pirms pirmās komandas un palīgvārds *beigas* – aiz pēdējās komandas.

Algoritmus, kura komandas izpilda tikai tādā secībā, kādā tās pierakstītas, sauc par *lineāriem* algoritmiem.

Algoritmus, kuru komandu izpildes gaitā jāizvēlas pie kāda nosacījuma izpildīšanās darīt vienu, bet pie nosacījuma neizpildīšanās darīt ko citu, sauc par *sazarotiem* algoritmiem.

Cikliski algoritmi. Lielākajā daļā algoritmu atsevišķas komandas jāatkārto daudzas reizes. Lai apzīmētu komandas, kas atkārtojas daudzas reizes, lieto speciālu konstrukciju – ciklu. Cikliskā struktūra satur nosacījumu, kuru izmanto, lai noteiktu, cik reižu attiecīgā komanda jāatkārto.

Pēc tam, kad algoritms ir uzrakstīts, to var izpildīt vienu vai vairākas reizes. To var izpildīt cilvēks vai kāda tehniska ierīce atkarībā no tā, kam algoritms ir paredzēts.

Par algoritma izpildītāju sauc cilvēku vai tehnisku ierīci, kas izpilda algoritmā norādītās darbības un iegūst rezultātu.

Lai izpildītājs varētu izpildīt algoritmu, tad algoritma darbībām ir jābūt uzrakstītām izpildītājam saprotamā valodā. Ja izpildītājs ir dators, tad algoritmu pieraksta programmas veidā kādā no programmēšanas valodām.

Algoritmus var pierakstīt vai attēlot visdažādākajos veidos:

1) pseidokodā

2) ar blokshēmām

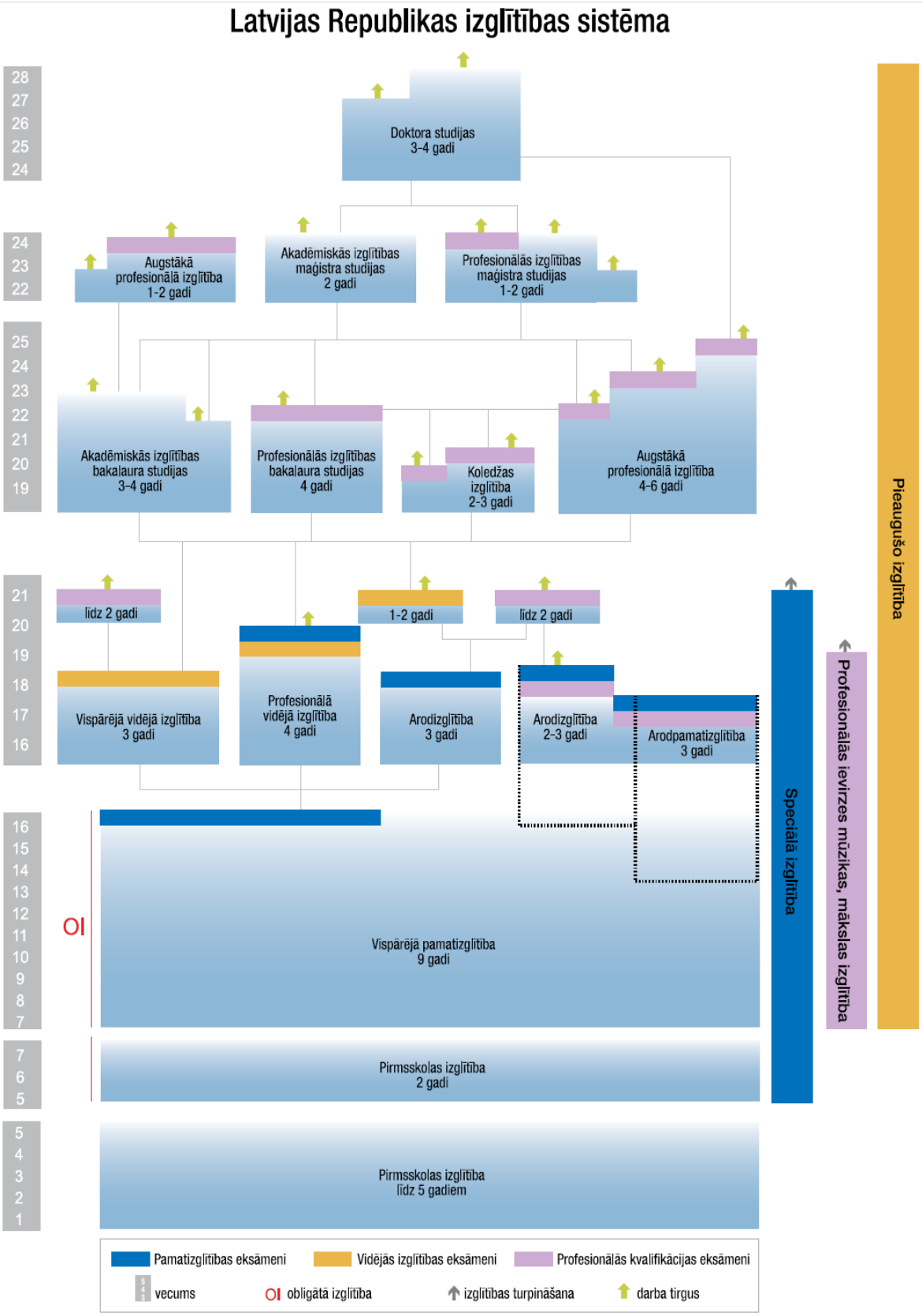
3) ar kādu ļoti striktu, nepārprotamu valodu, kas parasti ir kāda programmēšanas valoda.

Pseidokods ir pa vidu starp parastu valodu un programmēšanas valodu. Tā mērķis ir pietiekami loģiskā formā paskaidrot nepieciešamos darbības soļus, vienlaikus ignorējot lietas, ka cilvēkam attiecīgajā kontekstā ir pašsaprotamas.

Blokshēma ir vizuāls - grafisks veids, kā attēlot algoritma soļus - katrai darbībai piešķirot savu figūru un saistības starp šīm darbībām norādot ar bultiņām, kā arī paskaidrojumiem izmantojot tekstu. Savā ziņā to var uzskatīt par pseidokoda variantu.

1.1. Mācību priekšmeta „Informātika” saturs Latvijā

Izglītības sistēma Latvijā ir sadalīta vairākos posmos: pirmskolas izglītība, pamatskolas, vidējā izglītība un augstākā izglītība. Mācību priekšmets informātika tiek mācīts pamatskolā trīs gadus un divus gadus vidusskolā. Augstskolās informātika tiek mācīta atkarībā no profesionālās ievirzes prasībām.



2.1.attēls. Latvijas Republikas izglītības sistēma [3]

Informātika ir salīdzinoši jauns mācību priekšmets, ko apgūst skolā. Informācijas komunikāciju tehnoloģijas attīstās ļoti strauji. Līdz ar to regulāri jāmainās arī mācību saturam.

Šobrīd pamatskolā mācību priekšmeta „Informātika" obligāti apgūstamo saturu nosaka LR Ministru kabineta 2006. gada 19. decembra noteikumi Nr.1027 „Noteikumi par valsts standartu pamatizglītībā un pamatizglītības mācību priekšmetu standartiem"

Mācību priekšmeta „Informātika" apguves mērķis, kas definēts mācību priekšmeta standartā ir" sekmēt izglītojamā zināšanu pilnveidošanu un praktisko prasmju attīstīšanu moderno informācijas un komunikācijas tehnoloģiju lietošanā informācijas iegūšanai, apstrādei un analīzei, kas nepieciešama daudzveidīgās dzīves situācijās un citu mācību priekšmetu apguvē"[15].

Mācību priekšmeta standartā noteiktais mērķis norāda uz to, ka skolā iegūtajām zināšanām un prasmēm ir jābūt noderīgām reālajā dzīvē. Tās ir jāspēj izmantot citu mācību priekšmetu apguvē, meklējot un apstrādājot informāciju, strādājot ar dažādām programmām. Mūsdienās darba tirgus izvirza ļoti augstas prasības datorprasmēm.

Mācību priekšmeta „Informātika" standartā noteiktais mācību saturs atbilst Eiropas datorprasmes sertifikāta ieguves prasībām. Skolas uzdevums – nodrošināt standartā noteikto zināšanu un prasmju kvalitatīvu apguvi.

Latvijā informātika kā mācību priekšmets pamatizglītības standartā iekļauta no 5. – 7.klasei. Pamatskolā māca trīs gadus. Pamatizglītības standartā uzsvars tiek likts uz biroja lietotņu lietošanas apmācību un prasmēm strādāt ar dokumentiem. Kā praktiskais un pētnieciskais darbs tiek noteiktas prasmes:

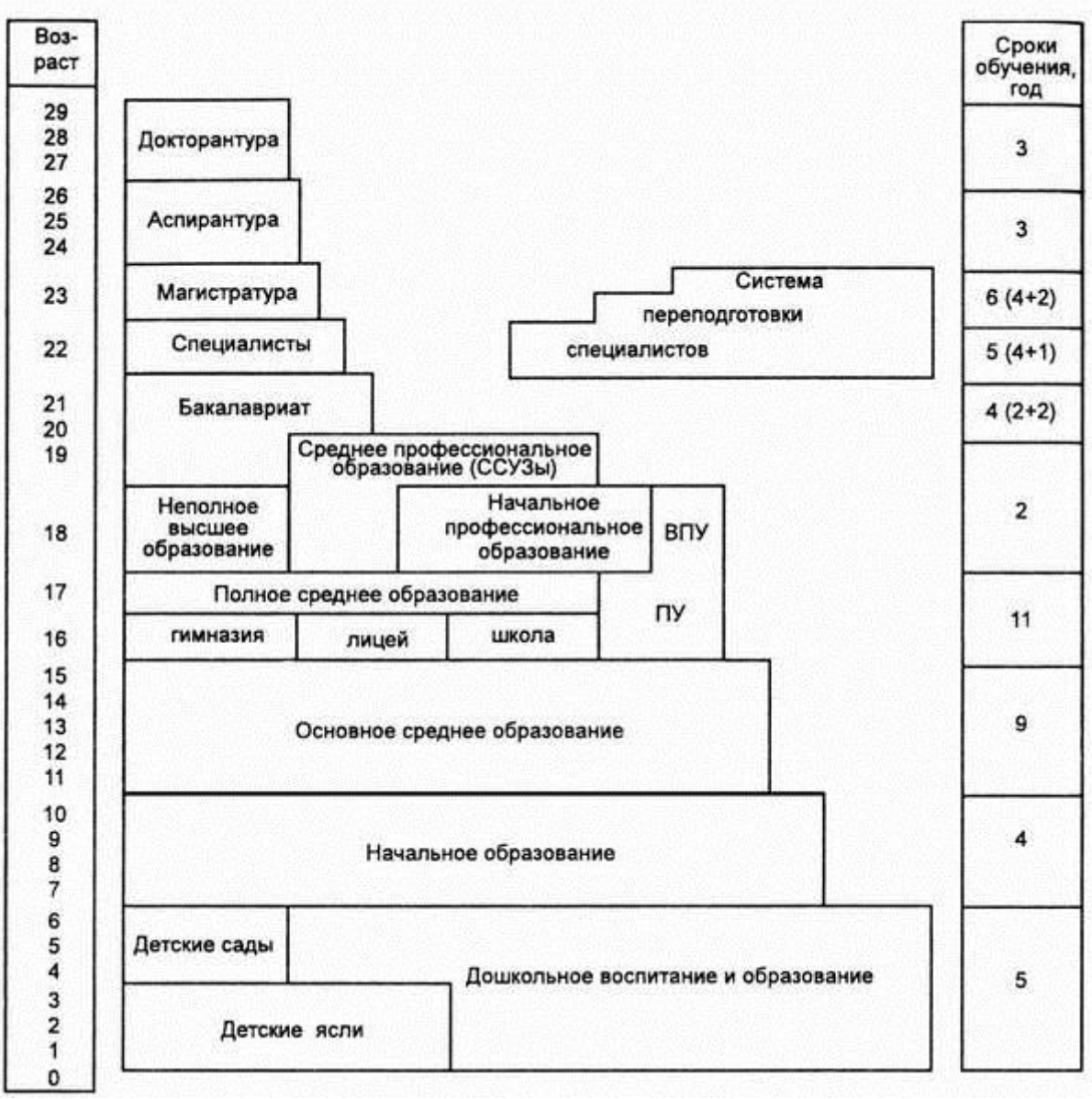
- datora lietošana un rīkošanās ar datnēm,
- attēlu apstrāde,
- teksta apstrāde,
- izklājlapu (rēķintabulu) apstrāde,
- prezentācijas materiālu sagatavošana un demonstrēšana,
- informācijas ieguves un komunikācijas līdzekļu izmantošana.

Jēdzienu „algoritms" pamatizglītības standartā ir paredzēts apskatīt līmenī – zina jēdzienu algoritms. Jēdziena izpratne un prasmes pielietot zināšanas iekļauts vidējās izglītības mācību priekšmeta „Informātika" standartā.

1.2. Mācību priekšmeta „Informātika" saturs Krievijā

Obligātās izglītības saturu Krievijā nosaka Federālais Valsts vispārējās pamatizglītības standarts. 2010.gadā pieņemtajā standartā ir noteiktas jomas: filoloģija, matemātika un informātika, dabaszinības (fizika, bioloģija, ķīmija), mākslas, tehnoloģijas, fiziskā kultūra un

sabiedrības mācība, kulturoloģija.[10] Informātika Krievijā tiek mācīta no 7.-9.klasei viena stunda nedēļā. Pavisam 105 stundas.



2.2.attēls. Krievijas izglītības sistēma [10]

Informātikas pamatizglītības standartā noteiktas iegūstamas zināšanas un prasmes: informatīvās un algoritmiskās izpratnes veidošana, jēdzienu „informācija", „algoritms" īpašību izpratne. Algoritmiskās domāšanas veidošana, kas nepieciešama profesionālajā darbībā mūsdienu sabiedrībā. Attīstīt prasmes algoritma korekta pieraksta izpildei, lai veiktu konkrētas darbības. Veidot zināšanas un prasmes par:

- zināt algoritma struktūru;
- izprast algoritma veidus (lineārs, sazarots, ciklisks);
- iepazīties ar vienu no programmēšanas valodām;

- prasmes datu apstrādē un informācijas strukturēšanā.

Izmantojot nepieciešamo programmatūru; veidot prasmes un iemaņas darbā ar datoru, zināšanas par drošu darba vidi un ētiku internetā.

Pamatizglītības standartā tiek noteiktas tikai vadlīnijas, detalizētu izglītības saturu nosaka Izglītības iestādes pamatizglītības paraugprogramma.

Paraugprogrammā ir 4 sadaļas:

- informācija un tās attēlošanas veidi,
- algoritmu pamati,
- programmatūras sistēmu un pakalpojumu pielietojums,
- darbs informācijas telpā.

Sadaļā „Informācija un tās attēlošanas veidi" paredz iegūstamās prasmes informātikas terminu lietošanā, informācijas pierakstīšana binārajā kodā, kodēšanā un dekodēšanā un citas saistītas ar informācijas attēlošanu.Algoritmiskās kultūras pamati paredz iegūstamās prasmes un zināšanas par algoritmiem.

Beidzot 9.klasi jāzina terminus algoritms un algoritma izpildītājs, prot patstāvīgi sastādīt un izpildīt vienkāršus algoritmus saistītus ar algoritmiem saistītus uzdevumus. Veidot modeļus dažādām ierīcēm un objektiem, kā izpildītājiem, aprakstīt šo izpildītāju sistēmas komandas.Izprast terminu "algoritms", zināt galvenās algoritmu īpašības (instrukciju kopums, viennozīmīgums, formalitāte, noteiktība, diskrētība, galīgums un efektivitāte).Prot sastādīt lineārus algoritmus un uzrakstīt to izvēlētā programmēšanas valodā. Prot sastādīt vienkārša uzdevuma algoritmu, izmantojot sazarotus un cikliskus algoritmus. Iepazīstas ar rindas, koka un grafa pielietojumu.

Apgūstot programmatūras sistēmu un pakalpojumu pielietojumu, tiek paredzēts apgūt pamatiemaņas un prasmes darbā ar failu sistēmu, teksta redaktoru, izklājlapām, pārlūkprogrammām, meklētājprogrammām, vārdnīcām un elektroniskajām enciklopēdijām. Paredz apgūt zināšanas, prasmes un iegūst pieredzi, lai strādātu pamata līmenī ar dažādām programmatūras sistēmām un pakalpojumu veidiem, spēj aprakstīt, kā darbojas šīs sistēmas, izmantojot atbilstošu terminoloģiju.

Iepazīstas ar audiovizuālo datu apstrādi, prot izveidot teksta dokumentus, grafiskus darbus, sagatavot prezentācijas materiālus. Iepazītas ar matemātiskās modelēšanas un datoru mūsdienu zinātniskās un tehniskās pētniecības piemēriem bioloģijā, fizikā, medicīnā, aviācijā, kosmonautikā un citās nozarēs.Darbs informācijas telpā paredz zināšanas par datortīkliem un internetu, pētīt, kā attīstās informācijas komunikāciju tehnoloģijas.

1.3. Mācību priekšmeta „Informātika” saturs Anglijā

Izglītība Anglijā tiek dalīta 6 posmos:

- pirmskolas izglītība līdz 6 gadiem;
- 1.posms – 6 – 7 gadi;
- 2.posms – 8 – 11 gadi;
- 3. posms - 12 – 14 gadi;
- 4. posms – 15 – 16 gadi;
- 5. posms – 17 – 18 gadi.

Izglītības obligātais saturs paredz noteiktas zināšanas un prasmes katrā no vecumposmiem. Mācību mērķis ir augstas kvalitātes izglītība skaitļošanas jomā, kas palīdz labāk izprast un mainīt pasauli ar skaitļošanas domāšanas palīdzību. Lai to īstenotu, nepieciešams attīstīt loģisko domāšanu un precizitāti, apvienojot ar radošumu un stingrību. Datorzinātne palīdz saprast gan dabīgas, gan mākslīgas sistēmas un ir būtiski saistīta ar matemātiku, zinātni, dizainu un tehnoloģiju. [9]

Nacionālās izglītības programmas skaitļošanas novirziena mērķis ir nodrošināt, lai visi skolēni:

- saprastu un pielietotu fundamentālas datorzinātnes pamatprincipus, ieskaitot loģiku, algoritmus, datu reprezentāciju un saziņu;
- var analizēt problēmas datortehnikā un no praktiskās pieredzes rakstīt programmu, lai problēmas atrisinātu;
- var novērtēt un pielietot IT zināšanas kopā ar jaunām un nepazīstamām tehnoloģijām, lai analizētu un risinātu problēmas;
- būtu atbildīgi, kompetenti, pārliecināti un radoši informācijas un komunikāciju tehnoloģiju lietotāji.

Katra posma beigās skolniekam būtu jāsaprot, jāspēj pielietot prasmes, kuras tik apgūtas noteiktajā posma un ir svarīgas noteiktajai programmai.

1.posms

Skolēnam iegūstamās zināšanas un prasmes:

- izprast, kas ir algoritmi, kā tie tiek pielietoti programmās uz digitālām ierīcēm un kā programmas izpilda instrukcijas;
- uzrakstīt un pabaudīt vienkāršas programmas;
- izmantot loģisku domāšanu un paredzēt vienkāršu programmu uzvedību;
- organizēt, saglabāt, manipulēt un atgūt datus dažādos digitālos formātos;
- droši un ar cieņu sazināties tiešsaistē, ievērot personīgas informācijas privātumu.

2.posms

Skolēnam iegūstamās zināšanas un prasmes:

- prot izveidot un uzrakstīt programmu, kas var paveikt noteiktu uzdevumu ar mērķi, kontrolēt un simulēt fizisku sistēmu;
- izmantot secību, atlases un atkārtošanas programmu, lai strādātu ar mainīgajiem un dažādām ievades, izvades formām, izvēlēties atbilstošu ievadi un paredzēt iznākumu testa programmām;
- izmantot loģisko domāšanu, lai izskaidrotu vienkāršus algoritmus, noteikt un izlabot kļūdas algoritmos un programmās;
- saprast datortīklus, ieskaitot internetu, kā tie piedāvā vairākus servisus, tādus kā globālais tīmeklis un iespējas, ko tas piedāvā saziņai un sadarbībai;
- aprakstīt kā interneta meklēšanas programmas atrod un sakārto datus, efektīvi izmantot meklētājus, novērtēt digitālo saturu, cienīt citus un citu intelektuālo īpašumu, izmantot tehnoloģijas droši un atbildīgi;
- izvēlēties un izmantot vairākas programmas uz dažādām ierīcēm, lai paveiktu noteiktus mērķus, ieskaitot datu un informāciju savākšanu, analīzi, izvērtēšanu un prezentēšanu.

3.posms

Skolēnam iegūstamās zināšanas un prasmes:

- dizains, izmantošana, skaitļošanas abstrakciju izvērtējums, kura modelē reālas pasaules problēmas un fiziskās sistēmas;
- saprast vismaz divus svarīgus algoritmus katrai kārtošanas un meklēšanas programmai, izmantot loģisko domāšanu, lai izvērtētu dažādus algoritmus, lai risinātu problēmu;
- izmantot divas vai vairāk programmēšanas valodas, lai risinātu vairākas skaitļošanas problēmas, izmantot tādas datu struktūras, kā tabulas vai masīvi, izmantot procedūras, lai rakstītu modulāras programmas, katru savai procedūrai, izskaidrot kā tas strādā un pārbaudīt;
- saprast vienkāršu Būla loģiku (kā UN, VAI, NE), izmantot, lai noteiktu kuras programmas daļas izmanto Būla loģiku. Novērtēt kā meklētāji izvēlas un sakārto rezultātus;
- saprast aparatūras un programmatūras komponenti, kas veido tīklu datorsistēmu, kā tie darbojas kopā, kā ietekmē izmaksu un sniegumu, izskaidrot kā internets strādā, izprast kā datori var novērot un kontrolēt fizisku sistēmu;

- izskaidrot kā instrukcijas ir saglabātas un izpildītas datorsistēmā;
- izskaidrot kā ar dažādiem datu tipiem var manipulēt ar digitāliem binārajiem skaitļiem, tekstu, skaņu un bildēm;
- iesaistīties radošos projektos, kuri iesaista izvēli izmantojot un kombinējot dažādas programmas, ieteicams ar vairākām ierīcēm, lai sasniegtu mērķus, iekļaujot datu savākšanu un apstrādi;
- radīt, izmantot, pārskatīt digitālo informāciju un saturu ar domu radīt intelektuālo īpašumu.

Algoritmu mācīšanai jau jaunāko klašu skolēniem dažādās pasaules valstī piešķir arvien lielāku nozīmi. Jau no pirmās klases bērniem jāapgūt datorpogrammēšanu un citas tehnoloģiju zināšanas, lai valsts varētu attīstīsies, uzskata igauņi[13]. Igaunijā ar īpaši izveidota fonda atbalstu skolas mērķtiecīgi apgādātas ar datortehniku un atbalstītas inovāciju projektos, tā kā programmēšana jau pirmajās klasēs ir nākamais loģiskais solis, uzskata fonda pārstāvji.

Krievijā algoritmu mācīšana ir iekļauta matemātikas stundās no 2.klases, algoritmu mācīšana tiek apskatīta arī valodas stundās pētot sintaksti gramatikā, līdz ar to no 7.-9.klasei sāk apgūt programmēšanas valodas.Savukārt Latvijas

Informācijas un komunikācijas tehnoloģijas asociāciju pēc nozarē notikušas plašas diskusijas par igauņu iniciatīvu nolēmusi pievērst lielāku uzmanību izglītības sfērai. Jau tuvākajā laikā tikšot sludināts konkurss par labāko skolotāju, kas motivējis savus skolēnus datorgudrības apgūt dziļāk[5].

Latvijas Informācijas un komunikācijas tehnoloģijas asociācija rosina no 5.klases ieviest jaunu izvēles mācību priekšmetu *Datorika*, kas ietvertu skolēnu vecumam atbilstošu algoritmikas un programmēšanas pamatu apguvi, kā arī vidusskolā matemātikas, dabaszinātņu un tehnikas virzienā obligāto/obligāto izvēles priekšmetu sarakstu papildināt ar mācību priekšmetu *Programmēšanas pamati*[5].

2. VIZUĀLĀS PRORAMMĒŠANAS VALODAS

Šodien arvien biežāk runā par to, ka dzīvojam informatizācijas pasaulē, ar to saprotot globālu informatizāciju. Strauji notiek arī izglītības informatizācija: vienota skolēnu datu bāze, e-klase. Izglītības iestādēs ienāk arvien vairāk informācijas tehnoloģijas – mobilā klase ar aprīkojumu ķīmijas, bioloģijas un fizikas mācīšanai, interaktīvās tāfeles, datu kameras un balsošanas pultis. Mācību process kļūst arvien interaktīvāks. Informātika tiek mācīta katrā skolā, daudzās skolās arī programmēšanas pamati, datorzinību pamati tiek piedāvāti ārpusstundu nodarbībās.

Algoritmu apguve veicina bērna garīgo attīstību un loģiskās domāšanas attīstību. Algoritmu pasniegšanu var organizēt dažādi. Var dot gatavus algoritmus, kurus skolēns izpēta, iemācās, un pildot vingrinājumus nostiprina zināšanas. Bet var organizēt mācību procesu tā, lai algoritmi paši „atvērtos", veidotos. Šis veids ir laikietilpīgs, bet no didaktikas viedokļa – ļoti vērtīgs.

Algoritmu mācīšanai pamatskolā var ir mantot dažādas programmas: *Alise, Scratch, Waterbearlang, Arduino, Logo, Turtle Art, Lego Mindstorm NXT* un citas.

Diplomdarbā sīkāk tiek apskatītas programmēšanas vides *Alise, Scratch, Lego Mindstorm NXT*. *Alise* un *Scratch* programmēšanas princips ir līdzīgs: komandas tiek ievilktas, objekta kurstība tiek orientēta uz tā koordināšu maiņu vai tiek vadīti izmantojot klaviatūru.

Vispirms jāsāk ar algoritmu pamatkonstrukciju apguvi.

2.1. Algoritmu pamati

Algoritmus var pierakstīt vai attēlot visdažādākajos veidos:

1) pseidokodā

2) ar blokshēmām

3) ar kādu ļoti striktu, nepārprotamu valodu, kas parasti ir kāda programmēšanas valoda.

Par algoritmiem sīkāk aprakstīts 1.nodaļā un 3.pielikumā.

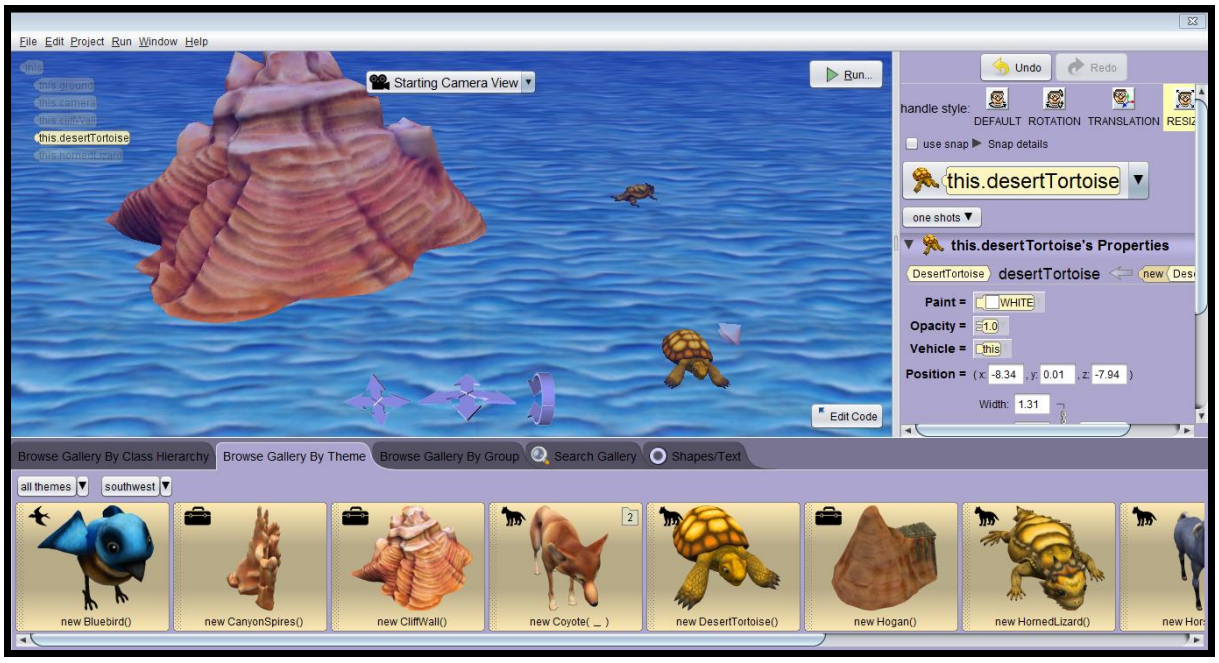
Programmēšanas elementus var sākt mācīt pamatskolā un pat jau sākumskolā. Kā Ievadkurss varētu būt algoritmu mācīšana. Izveidotas daudz dažādas datorprogrammas, kuras dod iespēju veidot atraktīvas un ar praktiskas darbošanās pielietojumiem nodarbībās algoritmu apguvē. Šodien, informāciju tehnoloģiju laikmetā ir nepietiekami būt tikai programmu lietotājam, vairāk vai mazāk vajadzētu tiekties uz izstrādātāju. Bērniem skolas vecumā programmēšana šķiet ļoti saistoša, tāpēc katram skolēnam vajadzētu vismaz vispārīgos vilcienos orientēties programmēšanas principos.

Informātikas stundās skolā māca pamatā lietojumprogrammas. Algoritmus un programmēšanas pamatus māca vidusskolas klasēs un tikai skolās, kuras ir izglītības programmas mācību plānā to iestrādājušas. Lielai daļai skolēnus šis kurss sagādā lielas grūtības. Situācija varētu uzlaboties, ja programmēšanas elementi tiktu iekļauti mācību procesā agrāk – pamatskolā vai pat sākumskolā. Tas pamatā mainītu domāšanu, attīstītu loģisko, strukturēto domāšanu. Liela nozīme izglītībā ir dažādu projektu izstrādei, kas arī ir radošā darbība. Turklāt algoritmu apguve – algoritma patstāvīga veidošana, formulēšana, tas arī ir radošs process. Līdz ar to algoritmu apguve skolā veido ne tikai strukturēto, bet attīsta arī radošo domāšanu.

2.2. Alise

Alice ir 3D programmēšanas vide, kas padara animāciju veidošanu ļoti vieglu, lai varētu spēlēt interaktīvas spēles vai video ar ko padalīties internetā. *Alice* ir orientēta darbam ar objektiem un vienkāršu skriptu pierakstam. Programmēšanas vidē *Alice* lietotājs var kontrolēt objektu izskatu un uzvedību. Skriptu izpildes laikā, objekti atbild uz peles un klaviatūras dotajām komandām. *Alice* ir veidota ar *Python* valodas palīdzību un izmanto vairākas *Python* funkcijas [4]. *Alice* ir brīvi pieejams apmācīšanas instruments radīts, lai kā pirmais solis darbā ar objektu orientētu programmēšanu. Tas atļauj iemācīties fundamentālus programmēšanas pamatus veidojot animētas filmas un vienkāršas video spēles.

Alice ir interaktīva lietotāja saskarne, kur programmas teksta veidošanai izmanto „vilkt un nomest" metodi. Instrukcijas atbilst programmēšanas valodu Java, C++ un C#.standarta sintaksei.



2.3. attēls. *Alise* vide

Alice projekta mērķis ir radīt iesācējiem viegli saprotamu un interesantu 3D vidi, lai izpētītu 3D objektu uzvedību. 3D modeļus objektiem, kā dzīvnieki, mašīnas utt. piepilda *Alice* pasauli.

Alice piedāvā vairākas iebūvētas komandas, kopumā komandas var sadalīt divās kategorijās:

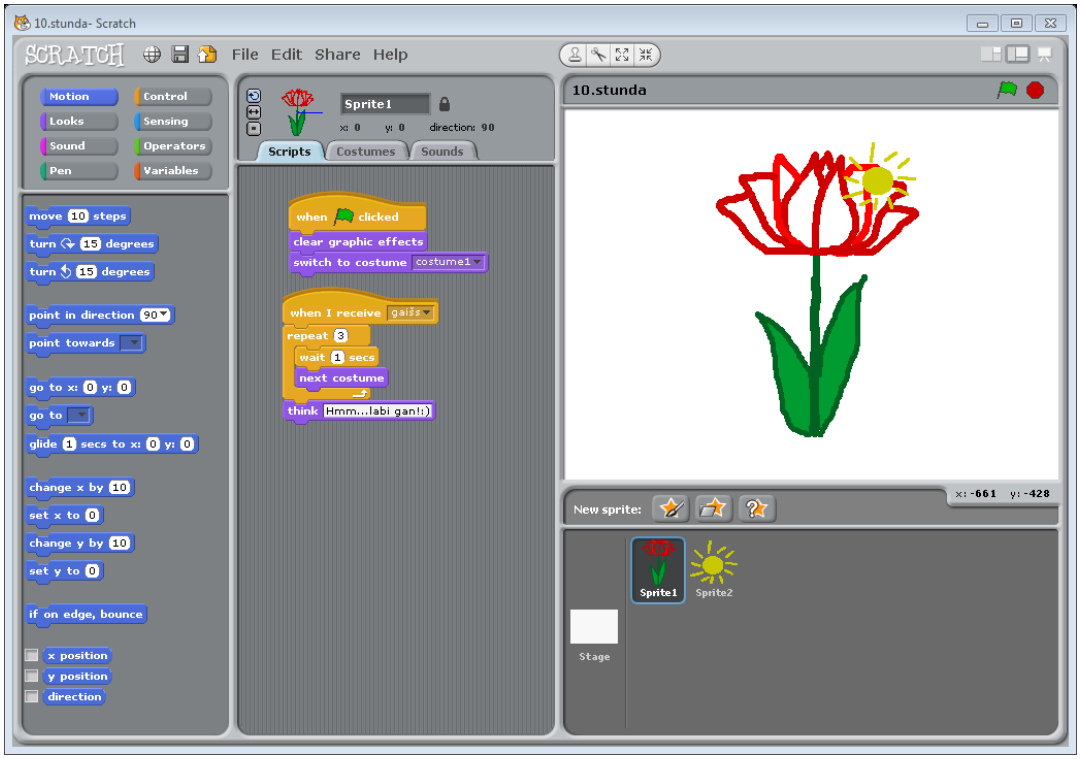
- norāda objektam kustības veidu,
- maina objekta fiziskas īpašības.

Alice ir programmēšanas valoda iesācējiem, lietotājs var uzreiz redzēt kā viņu animēta programma strādā. Tas palīdz labāk saprast dažādo programmas valodas sintaksi.

2.3. *Scratch*

Scratch ir vienkārša un interesanta vizuālas programmēšanas vide ar kuras palīdzību var bērniem mācīt vienkāršus algoritmus [14]. Ar *Scratch* palīdzību var viedot animācijas, spēles, audiovizuālus mākslas darbu. *Scratch* sevī ietver programmēšanu, grafikas elementus, un situācijas modelēšanas iespējas. Ar programmas palīdzību var izveidot animācijas un vienkāršas spēles.

Scratch sintakse ir līdzīga kā *Pascal*. Komandas, kas *Pascal* rakstītas skriptu veidā, *Scratch* ir dotas bloku veidā, turklāt pārskatāmībai ir izvietotas attiecīgās sadaļās. Līdz ar to, *Scratch* apguve, var nopietni palīdzēt izprast algoritmu būtību, programmēšanas pamatus, kas var palīdzēt nopietnākām programmēšanas studijām.



2.4.attēls. *Scratch* vide

Programmas vide ir ļoti vienkārša, tāpēc iespējams programmēt jau sākumskolas vecumam jau no gadiem 10. Strādājot šajā vidē bērniem ir iespēja attīstīt daudzas 21.gadsimtā būtiskas prasmes:

- radošums;
- prasme mācīties, pašmācība;
- situācijas analīze;
- tehnikas ātra apguve;
- projektēšana;
- efektīva komunikācija.

Kaut *Scratch* izmanto programmēšanas idejas, pēc būtības tā ir kā spēlēšanās ar *Lego* klucīšiem, tikai augstākā līmenī. Lietotājs var izveidot programmu no blokiem, tāpat kā saliekot konstrukciju no klucīšiem.

Scratch ir izteikti objektorientēta valoda. Katram objektam ar savi atribūti -atrašanās vieta, virziens, izskats. Katram objektam var vizuāli uzprogrammēt dažādus skriptus, vai metodes, kuras izpildās konkrētu notikumu gadījumā, piemēram, ja tiek nospiesta sākuma poga, notiek darbības ar peli, ja tiek piespiests taustiņš, tiek saņemts ziņojums un citas darbības.

Mācoties *Scratch* un strādājot šajā vidē, lietotājs var apgūt:

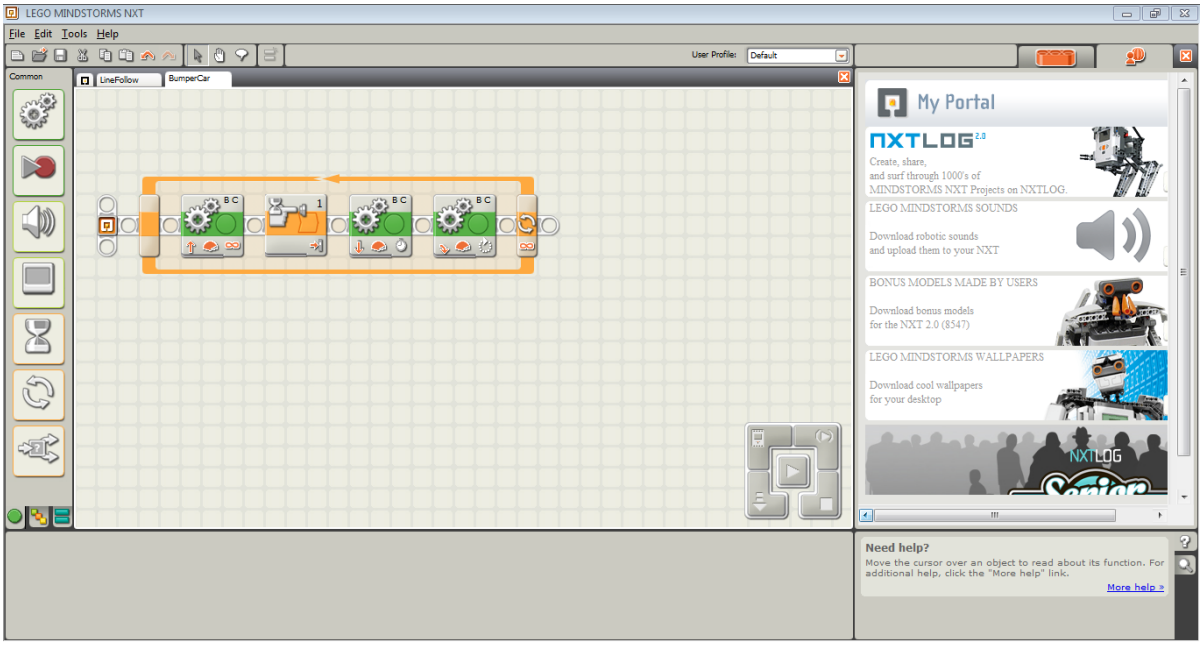
- algoritma un programmas nozīmi;
- patstāvīgi pieņemt lēmumu;
- patstāvīgi sastādīt algoritmu;
- izpētīt cikla darbības funkcionalitāti;
- nosacījuma operatorus.

Algoritmu un programmēšanas pamatus izmantojot *Skretch* iespējams mācīt ārpusstundu nodarbībās, pulciņos, fakultatīvajās nodarbībās vidējā skolas vecuma bērniem. Algoritmu un programmēšanas pamatu mācīšana tik agrā vecumā varētu likt pamatus nopietnāku programmēšanas valodu apguvē, iespēju piedalīties tematiskajās olimpiādēs un konkursos. Strādājot ar bērniem nedrīkst nepamanīt apdāvinātos un talantīgos bērnus, kuri ir apdāvināti un kuriem ir padziļināta interese par *Skretch*. Tāpēc nozīmīgi ir izstrādāt metodiku pamata līmenim un izstrādāt uzdevumus laboratorijas darbus speciāli apdāvinātajiem skolēniem.

2.3. *Lego Mindstorm NXT*

Lego ir viena no pasaulē populārākajām rotaļlietām, no tiem var izveidot visneiedomājamākās lietas – sākot ar mašīnām un beidzot ar robotiem. *Lego* ir dāņu kompānijas *"Lego Group*" ražotas rotaļlietas, kuras tautā bieži sauc arī par *Lego* klucīšiem. *Lego* klucīši ir šīs kompānijas galvenais produkts, kuri tiek izmantoti, lai izveidotu dažādas minifigūriņas, kā arī dažādas detaļas. Kompānija ražo arī speciālas detaļas, no kurām var izveidot vieglās un smagās mašīnas, lidmašīnas, vilcienus, ēkas, pilis, skulptūras, kuģus, kosmosa kuģus un pat reāli darbojošos robotus. Darbs ar konstruktoru rada visus priekšnosacījumus personības harmoniskai intelektuālai attīstībai. Lai no bērnības iemācītu maksimāli izmantot intelektuālo potenciālu, *Lego* ir radījis *Lego Mindstorms NXT*. Tas piedāvā unikālu iespēju reālu konstruēšanu apvienot ar neierobežotām programmēšanas iespējām.[8]

Konstruktora *Lego Mindstorm NXT* izmantošanai mācību procesā nepieciešamā materiāli tehniskā bāze ir: *Lego Mindstorm NXT* konstruktors, personālie datori ar grafikās programmēšanas vide Mindstorms NXT.



2.5. attēls. *Lego Mindstorm NXT* vide

Lego Mindstorm NXT izmanto 32-bitu datoru, kas kontrolē NXT bloku, interaktīvu Servo motoru, skaņu, ultraskaņu un citus sensorus, Bluetooth komunikācijas un vairākas lejuplādes iespējas. Uz ikonām bāzēta *Lego Mindstorm NXT* programma ir veidota uz LabVIEW™ programmas no National Instruments.

2.4. Vizuālās programmēšanas *Alise, Scratch, Lego Mindstorm NXT* salīdzinājums

Vizuālās programmēšanas vides ir interesantas un saistošas ne tikai ar iespējām izveidot animāciju, bet pats programmēšanas process ir interesants un vizuāli saistošs.

2.1.tabula.Alise, Scratch, Lego Mindstorm NXT salīdzinājums.

Kritēriji	*Alise*	*Scratch*	*Lego Mindstorm NXT*
Lietotāja saskarne	interaktīva, izmanto „vilkt un nomest" metodi	interaktīva, izmanto „vilkt un nomest" metodi	interaktīva, izmanto „vilkt un nomest" metodi
Aizņem vietu uz cietā diska	500 Mb	33Mb, iespējams strādāt tiešsaistē	300Mb
Vides vizuālais izskats	Krāsaina un interesanta	Vienkārša	Vienkārša
Pieejami materiāli	Maz un angļu valodā	Ļoti daudz krievu un angļu valodās	Ļoti daudz krievu un angļu valodās
Pieejamie uzdevumi	Maz un angļu valodā	Ļoti daudz krievu un angļu valodās	Ļoti daudz krievu un angļu valodās
Uztveramība	Ļoti vienkārša, ātri uztverama	Gandrīz vienkārša, programmēšanas valoda vidēji sarežģīta	Programmēšanas valoda vidēji sarežģīta.
Vērtējums punktos (1-3)	2	2	3

Katrai programmai ir parametri, kuri noska kādu programmēšanas vidi izvēlamies. Tie ir kā vizuālie, tā arī tehniskie parametri, piemēram, prasības pēc brīvas vietas atmiņā uz cietā diska.

Aloritmu apguvei var izmantot dažādas programmēšanas valodas. *Scratch* un *Alise* ir vienkāršas un interesantas vizuālas programmēšanas vides, ar kurām var ieinteresēt skolēnus programmēšanas pamatu apguvē. Tās ir vienkāršas un saprotamas, paredzētas bērniem jau no 10 gadu vecuma.

Lego Mindstorm NXT programmēšanas vide ir sarežģītāka, kā iepriekšminētās, turklāt programmas tekstu var pārbaudīt tikai uz konkrētas ierīces. Strādājot ar *Lego Mindstorm NXT* mācās ne tikai programmēt, bet arī projektēt, tas jau ir solis uz priekšu robotikas nozarē.

3. ALGORITMU APGUVES IESPĒJAS IZMANTOJOT VIZUĀLĀS PROGRAMMĒŠANAS VIDES

Pētījums tika veikts lai noskaidrotu programmēšanas valodu *Scartch, Alise, Lego Mindstorm NXT* mērķauditoriju: vai skolēni tiek galā ar uzdevumu, vai algoritma veidošana ir saprotama. Balstoties uz pētījumā veiktajiem novērojumiem un gūtajiem secinājumiem veidots mācību tematiskais plāns.

3.1. Pedagoģiskais pētījums

Pedagoģiskais pētījums veikts X skolā, piedaloties

25 7.klases skolēniem,

36 6.klases skolēniem,

30 5.klases skolēniem

un 20 skolēnu vecākiem un 4 bērniem 6-7 gadu vecumā.

Pētījums veikts projekta nedēļas ietvaros.

Pētījuma metode – eksperiments.

Eksperimenta priekšmets – algoritma apguve.

Eksperimenta objekts – programmu *Scartch, Alise, Lego Mindstorm NXT* izmantošana algotumu apguvē.

Pētījuma mērķis: iepazīstināt ar algoritmu apguves iespējām, izmantojot vizuālās programmēšanas programmas.

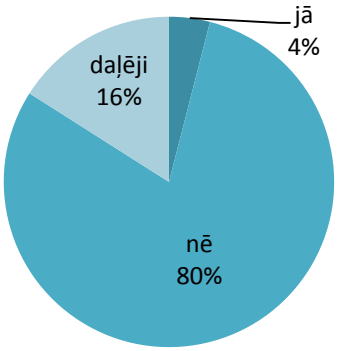
1.nodarbība. Grupa – pāris. Skolēni strādā pāros. Tiek pētīta programmas *Scratch* izpratne. Mērķauditorija 6. un 7.klases skolēni: 3 grupas 6.klases skolēni un 2 grupas 7.klases skolēni.

7.klases skolēni mācību programmas ietvaros ir iepazīstināti ar jēdzienu algoritmi, tāpēc strādājot ar *Scratch* netiek piedāvāti paraugi un instrukcijas. Programmas darbības principi tiek noskaidroti empīriskā ceļā, līdzdarbojoties pārī.

Vērojums: strādājot pa pāriem, skolēni ļoti aktīvi darbojas, ļoti aktīvs mācīšanas process, diskusijas, pareizā risinājuma meklējumi. Novērojot 12 grupas – pārus, varu secināt, ka 40 minūšu stundas laikā arī bez iepriekšējām zināšanām Scartch vidē var radīt animāciju. Gandrīz visas grupas izveidoja nelielas animācijas. Daļa skolēnu izmantoja iebūvēto bibliotēku, daži izmantoja zīmēšanas iespējas. Protams, lai izveidotu animāciju vai spēli ar konkrētu sižetu, noteiktām prasībām ir vajadzīgas vairākas stundas un ievadlekcija par darbības principiem, algoritma uzbūvi un veidošanas posmiem.

Nodarbības beigās skolēni aizpilda anketas (1.pielikums), kurās atbild uz konkrētiem jautājumiem un izsaka savu viedokli un vērtējumu.

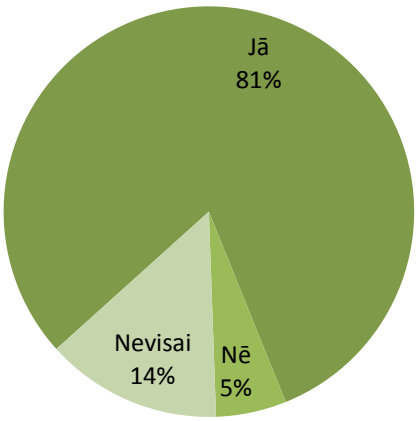
Visi skolēni apstiprināja, ka no sākuma bija neizpratne par to, kā var kaut ko izdarīt. Skolēni apstiprina, ka sākumā ir grūti izprast programmas vidi un lai izpētītu visas programmas piedāvātās iespējas un rīkus ir vajadzīgs daudz vairāk laika.



3.1.diagramma. Vai programmas izpratne sagādāja grūtības?

Skolēni anketās brīvā tekstā izteic savas domas par programmu. Apkopojot anketas, var secināt, ka skolēni kopumā ir apmierināti ar programmu, ar iespējamo rezultātu, atzīst, ka strādāt šajā vidē bija aizraujoši un interesanti, labprāt turpinātu darbu šajā vidē.

6. klases skolēniem tiek novadīta ievadlekcija 10. min. par animācijas veidošanu vizuālajā programmēšanas vidē *Scratch*, jo liela daļa skolēnu nezina jēdzienu algoritms. Programmēšana izmantojot blokus, kas satur programmas tekstu ir viegli saprotama. Skolēni, kuri ir radoši, ļoti ātri uztver programmas kodu un veiksmīgi tiek galā ar uzdevumu. Viena skolēnu grupa mācību stundā izveido nelielu spēli - objektu vada ar peli. Skolēniem ir ļoti liela interese par programmēšanas valodu un programmas iespējam. Par to liecina aptaujas rezultāti:



3.2.diagramma. Vai programmēšanas algoritms ir saprotams?

Skolēni apstiprina, ka ir izprotams programmas veidošanas algoritms. Skolēniem ir labas angļu valodas zināšanas un nekādas problēmas nesagādā darbs ar komandu blokiem, ir tikai jāplāno darbība.

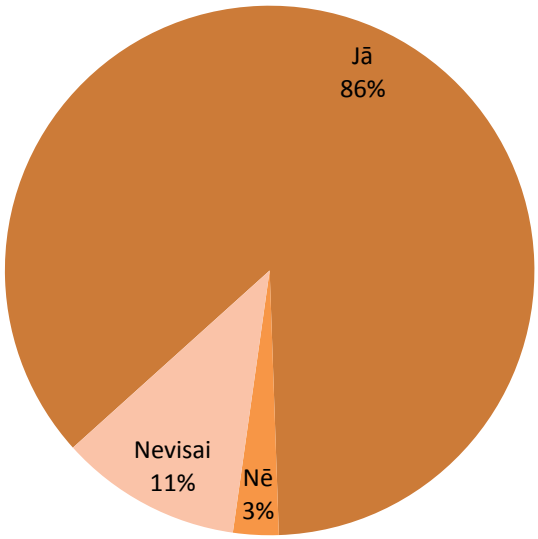
Skolēni pilnībā atbalsta programmēšanas apguves iespējas pamatskolā. Liela daļa skolēnu izprot loģiskās un strukturētās domāšanas pilnveides nozīmīgumu, kā iespēju tālākizglītībai.

2.nodarbība. Pāru darbs. Skolēni iepazīstas ar programmēšanas vidi *Alise*. Skolēnus aizrauj 3D vide un vienkāršās darbības ar objektiem. Objektus var brīvi pārvietot, palielināt, samazināt. Piedāvātā plašā bibliotēka ļauj vaļu fantāzijām. Skolēni iekārto mājokļus, darbojas jūras dzelmē, kosmosā. Skolēni atzīst, ka tā ir lieliska iespēja izpausties, lieliska iespēja fantazēt un likt objektiem pārvietoties.

Darbs pāros ir ideāls šādam eksperimentam – parasti viens ir izpildītājs, kas fiziski pilda darbu un otrs ideju radošais ideju ģenerētājs, darba intelektuālais vadītājs.

Darbs ar programmas tekstu ir līdzīgs kā programmēšanas valodā *Scretch*. Tā kā skolēni bija tikko strādājuši *Scretch* vidē, tad sastādīt objektu kustības algoritmu nesagādā grūtības. Skolēni atzīst, ka *Alise* vide ir vienkāršāka, vieglāk saprotama un no dizaina viedokļa pievilcīgāka, krāsaināka.

Anketēšanas rezultāti pierāda, ka skolēniem šķiet interesanta programmēšanas vide *Alise*, programmas valoda saprotama, vizuālais izskats interesants, saprotams, vienkārši orientēties. Visas nepieciešamās komandas ir viegli atrodamas.



3.3.diagramma. Vai programmas algoritms ir saprotams?

Skolēni novērtē programmas iespējas un izteic vēlēšanos turpināt strādāt ar šo programmu arī nākošajās stundās, vai ārpusstundu nodarbībās, Izteic vēlēšanos izveidot multfilmu vai vienkāršu spēli.

3.nodarbība. Vienkārša algoritma veidošana ar *Lego Mindstorm NXT*.

Nodarbības tiek vadītas 5., 6., 7. klašu skolēniem, kā arī 4.klašu skolēniem kopā ar vecākiem un jaunākajām māsām un brāļiem. Skolēni strādā individuāli, katrs pie savas darba stacijas. Skolotājs visām klašu grupām novada ievadlekciju, demonstrējot *Lego Mindstorm NXT* robota iespējas. Vispārējos vilcienos parāda kā iesākt veidot algoritmu, izskaidro rīku nozīmi. Vidē *Scratch* un *Alise* programmas kods ir rakstīts teksta veidā, skolēnam, kas saprot angļu valodu nav problēmas salikt kopā loģisku algoritmu. *Lego Mindstorm NXT* programmu veido ar blokiem, un bloku darbības principus ir jāizprot, tāpēc vispārīgos vilcienos tie tiek izskaidroti.

Skolēni ātri uztver algoritma veidošanas principus. Reālais laiks nelielas programmas veidošanai ir apmēram 15 minūtes, jo vismaz 15 minūtes nepieciešamas programmas testēšanai un kļūdu analīzei. Skolotāja programmu ievada robotā, skolēns izstāsta kāds ir programmas mērķis, paskaidro katra algoritma soļa būtību un nozīmi programmas tekstā. Pēc nelielā stāstījuma notestējam programmu ar robota palīdzību un analizējam, vai robots dara to, ko skolēns bija gribējis ieprogrammēt. Ja programmas tekstā ir kļūdas, kopā visa klase analizē programmas kodu un cenšas atrast kļūdu (programmas teksts tiek projecēts). Kļūdu novēršana un atkārtota testēšana vienas mācību stundas laikā nav iespējama, tāpēc *Lego Mindstorm NXT* būtu izmantojams ārpusstundu nodarbībās, nevis mācību stundā.

Divu 4.klašu skolēniem nodarbība notiek kopā ar vecākiem. Nodarbības mērķis izpētīt, kā ar uzdevumu tiek galā vecāki un vecvecāki, vecāku lomu mācīšanās procesā. Skolēni labi tika galā ar uzdevumu. Vecāki arī bija ļoti ieinteresēti. Katrs izveidoja savu programmu, kura tika notestēta, atrastas kļūdas un kļūdas izlabotas, un vēlreiz notestēta. Tā kā nodarbība ilga divas astronomiskās stundas, tad pietika laika arī kļūdu labojumam. Nodarbības laikā vēroju vecāku un bērnu sadarbību. Sadarbība bija abpusēja: skolēni palīdzēja vecākiem un vecāki skolēniem.

Grūtības saprast programmas darbības algoritmu sagādāja gados vecākiem un jaunākiem: vecmammām, vectēviem un bērniem 6 - 7 gadu veciem. Pēc nodarbības intervēju vecākus un vecvecākus. Vecāki ir ieinteresēti un saprot robottehnikas nozīmību. Izrāda interesei par *Lego Mindstorm NXT* programmēšanu kā nopietnu ārpusklases nodarbību, kas ievērojami paplašina bērnu zināšanas, attīsta domāšanu. Vecāki apstiprināta to, jaunāko klašu skolēni nesaprata kā izveidot programmu patstāvīgi.

Aptaujāju skolēnus, kuri piedalījās eksperimentā novērtēt, kura programma patika labāk, ar kuru vieglāk strādāt, ar kuru no programmām interesantāk strādāt.

Lielākā daļa skolēnu dod priekšrocību programmēšanas videi *Alise*, tā piesaista tās grafika, interfeiss un programmēšanas saprotamība. Par *Lego Mindstorm NXT* interese liela, bet kopā ar konstruēšanu tas ir nopietns laikietilpīgs darbs vairākos etapos: konstruēšana, programmēšana, testēšana, ko nav iespējams vienas mācību stundas laikā paveikt. Programmas teksta veidošana skolēniem nesagādāja grūtības un patika. Programmu salīdzinājumā skolēni atzīst, ka *Scratch* interesēja vismazāk.

3.2. Metodiskie ieteikumi

Pedagoģiskajā pētījumā tika izmantotas trīs programmēšanas vides: *Scartch*, *Alise*, *Lego Mindstorm NXT*. Mācību materiāls izstrādāts tikai uz *Scartch* un *Lego Mindstorm NXT*. *Scratch* sintakse ir līdzīga kā *Pascal*, *Alise* instrukcijas atbilst programmēšanas valodu Java, C++ un C#.standarta sintaksei. Vidusskolās pamatā māca programmēšanas valodu *Pascal*, tāpēc izvēlējos mācību materiālā iekļaut *Scratch*.

Svarīgi ir vispirms apgūt algoritmu pamatjēdzienus un pamatkonstrukcijas, tikai pēc tam kādu no programmēšanas valodām.

Mācību materiāls izstrādāts trīs tēmām: algoritmi, *Scartch* un *Lego Mindstorm NXT*. Kopumā 35 nodarbībām: katrai tēmai paredzētas 10 stundas, 5 stundas projetu darbu izstrādei pēc izvēles (skatīt 3.pielikumu).

Izstrādāts mācību tematiskais plāns un un nodarbību tematiskais plānojums algoritmu un programmēšanas pamatu pulciņa nodarbībām.

Skolēnu vecums – 5.klase. Uzdevumi sastādīti ņemot vērā 5.klases skolēna zināšanas matemātikā.

Mācību programmas mērķis: iepazīstināt ar algoritmu veidošanas un to izpildes pamatprincipiem, vienkāršu programmu rakstīšanu.

Skolēnam iegūstamās prasmes un iemaņas:

1. darbības plānošana:
   1.1. projekta plāna sastādīšana zīmējuma, shēmas veidā;
   1.2. projekta plānošana tabulas veidā, objekti, to īpašības, objektu mijiedarbība;
   1.3. sadalīt uzdevumu pa apakšuzdevumiem.
2. darbs grupās, lomu un uzdevumu sadalīšana;
   2.1. projekta plāna sastādīšana;
      2.1.1. tēmas izvēle;

2.1.2. uzdevuma sadalīšana apakšuzdevumos;

2.1.3. objektu izvēle;

2.2. rezultāta analīze, secinājumi;

2.3. kļūdu meklēšana un novēršana;

2.4. darba atskaites veidošana;

2.5. darba publiska aizstāvēšana;

2.6. projekta tālākās attīstības iespēju iezīmēšana.

Mācību materiālā ievietoti autores veidotie uzdevumi, kā arī izmantoti skolotāju izstrādātie materiāli.

SECINĀJUMI

Hipotēze ir apstiprinājusies - algoritmu apguve, izmantojot vizuālās programmēšanas programmas, ir vienkārša un saprotama. Var izmantot mācību darbā stundās un ārpusstundu nodarbībās.

Algoritmu un programmēšanas pamatu mācīšana agrā vecumā var likt pamatus nopietnāku programmēšanas valodu apguvei un labāk sagatavot turpmākajām mācībām.

Algoritmu, programmēšanas un robotikas pamatu mācīšana pamatskolā ir solis pretī nākotnes izglītībai. Tas palīdzētu popularizēt robotikas un dažādu datorikas virzienu skolēnu zinātniski pētniecisko darbu izstrādē.

Darba pētnieciskā daļa apstiprināja skolēnu un skolēnu vecāku ieinteresētību algoritmu un programmēšanas valodu apguvē. Skolēni algoritmu un programmēšanas pamatu apguves iespējas pamatskolā uztver kā nopienu iespēju savu izglītību tālākā nākotnē saistīt ar informācijas tehnoloģijām.

IZMANTOTĀ LITERATŪRA UN AVOTI

1. Balode A. Programmēšanas pamati. Valoda Turbo Pascal. Apgāds Zvaigzne ABC, 2009 (10 lpp; 113. lpp)
2. Špona A. Čehlova Z. Prātniecība pedagoģijā „Izdevniecība RaKa", 2004
3. http://www.aic.lv/portal/izglitiba-latvija
4. http://www.*Alice*.org/index.php
5. http://www.diena.lv/latvija/zinas/likta-rosina-skolas-macit-datoriku-un-programmesanu-13976056
6. http://en.wikipedia.org/wiki/Education_in_England
7. http://lv.wikipedia.org/wiki/Algoritms
8. http://lv.wikipedia.org/wiki/*Lego Mindstorm NXT*
9. http://media.education.gov.uk/assets/files/pdf/n/national%20curriculum%20consultation%20-%20framework%20document.pdf
10. http://standart.edu.ru/catalog.aspx?CatalogId=2588
11. http://w3.ppf.lu.lv/studijas/Praktikums/Mindstorms%20apraksts.pdf
12. http://www.uzdevumi.lv/ExerciseRun/RunExercise?exerciseId=e9afcee2-6c97-47f4-af43-367f22b485da&parentType=VirtualSchool&parentId=2877
13. http://www.e-klase.lv/lv/zina/zinas/aktualitates/igaunijas-skolas-no-pirmas-klases-macis-programmesanu/
14. http://younglinux.info/book/export/html/266
15. Noteikumi par valsts standartu pamatizglītībā un pamatizglītības mācību priekšmetu standartiemValsts pamatizglītības standarts . Pieejams http://www.likumi.lv/doc.php?id=150407

PIELIKUMI

1.pielikums. Anketa

1.jautājums.	Vai saprotama programmas vide.	Jā	Nē	Nevisai
2.jautājums.	Vai programmas algoritms ir saprotams?	Jā	Nē	Nevisai
3.jautājums.	Vai programmas izpratne sagādāja grūtības?	Jā	Nē	Daļēji
4.jautājums	Vai patika strādāt šajā programmā?	Jā	Nē	Nevisai
5.jautājums	Vai jūs gribētu šo programmu apgūt ārpusstundas nodarbībās?	Jā	Nē	Nevisai
6.jautājums	Vau jūs ieteiktu savam draugam (draudzenei) šo programmu kā interesantu pašizglītošanās iespēju?	Jā	Nē	Nevisai

2.pielikums. Mācību tematiskais plāns.

Mācību tematiskais plāns

	Tēma	Stundu skaits
1	**Algoritmi**	**10**
2	**Scratsc izmantošana algoritmu apguvē**	**10**
3	***Lego Mindstorm NXT* Mindstorm izmantošana algoritmu apguvē**	**10**
4	**Projektu darbs**	**5**
	Kopā:	**35**

Nodarbību tematiskais plānojums

Tēma №	**Stundu skaits**	**Stundas tēma**	**Apgūstamie jēdzieni.**
	1. Algoritmi		
1	1	Drošības noteikumi datorklasē	Drošības noteikumi datorklasē.
2	1	Algoritmi	Algoritma jēdziens. Vienkārša algoritma stādīšana (lineāri algoritmi).
3	2	Algoritma shēma, lineāri algoritmi	Algoritma shēma. Blokshēmas. Algoritma sastādīšana.
4	2	Sazaroti algoritmi	Nosacījuma algoritma sastādīšana: nosacījums: «jā» vai «nē» algoritmā.
5	2	Cikliski algoritmi	Cikla izpildīšana algoritmā. Soli pa solim veidot blokshēmas.
6	2	Uzdevumu par algoritmiem risināšana	Lineāri un sazaroti algoritmi, cikla izmantošana algoritmā
	10		
	2. Algoritma pieraksts ar *Scratch*		
1	1	Iepazīšanās ar *Scratch* vidi	Spritu izmantošana
2	1	Objektu vadīšana	
3	1	Vienlaicīga un secīga darbību izpilde	
4	1	Nosacījumi un mainīgie	
5	1	Gadījuma skaitļi	
6	1	Zīmēšana ar *Scratch*	
7	1	Dialogs ar programmu	
8	1	Objektu un kostīmu veidošana	
9	1	Objektu bibliotēkas izmantošana	
10	1	Animācijas veidošana	
	10		
	3. ***Lego Mindstorm NXT* Mindstorms NXT projektēšana un programmēšana**		
1	1	Iepazīšanās ar *Lego Mindstorm NXT* NXT 2.0 vidi	Roboti, robotu nozīme ikdienā. *Lego Mindstorm NXT* NXT 2.0 vide.
2	2	Iepazīšanas ar lineāra algoritma	

		veidošanu NXT vidē.	
3	2	Robota pārvietošanās pa noteiktu trajektoriju	
4	2	Cikla realizācija NXT vidē	
5	3	Projektu veidošana, patstāvīgs darbs NXT vidē.	
	10		

3.pielikums. Izdales materiāls skolotājiem un skolēniem

Algoritmi, to veidošanas pamatprincipi

1., 2.nodarbība. Drošības noteikumi datorklasē. Algoritmi.

Skolēni tiek iepazīstināti ar drošības noteikumiem datorklasē.

Algoritms ir precīzs un nepārprotams priekšraksts jeb norādījums izpildītājam veikt kādu darbību virkni, lai sasniegtu norādīto mērķi vai atrisinātu uzdevumu. Algoritma pieraksts ir sadalīts precīzos soļos. Katrā solī tiek dots viens norādījums, kuru sauc par komandu.

Algoritma īpašības:

- algoritms norāda, kas jādara, lai atrisinātu kādu noteiktu uzdevumu.
- algoritmam ir jābūt: precīzam, nepārprotamam un efektīvam.
- algoritmam jābūt pierakstītam tā, lai dažādi izpildītāji to saprastu, neatkarīgi no izpildītāja.

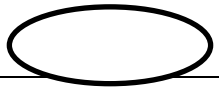
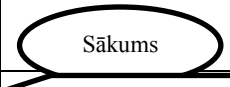
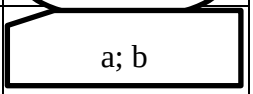
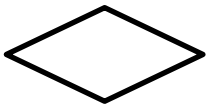
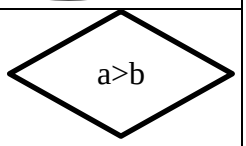
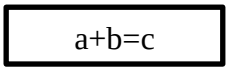
Algoritma izpildītājs. Katrs algoritms veidots noteiktam izpildītājam. Algoritma izpildītāji var būt cilvēki, kā arī datori, roboti un citas tehniskas ierīces. Algoritma pierakstam jābūt izteiktam jebkuram izpildītajam saprotamā formā.

Algoritma pieraksta metodes:

- vārdisks apraksts
- blokshēma,
- programma kādā programmēšanas valodā

Blokshēma. Par blokshēmu sauc algoritma loģiskās struktūras attēlojumu grafiskā formā, kurā katra algoritma darbība ir apzīmēta ar ģeometrisku figūru (bloku).

Blokshēmu elementi

Nosaukums	Raksturojums	Elements	Piemērs
Sākums, beigas	Algoritma sākums vai beigas		
Ievade	Datu ievade		
Izvade	Apstādes rezultātu parādīšana (izvade)		
Lēmuma pieņemšana,sazarojums	Pārbauda, vai dotais nosacījums izpildās vai neizpildās		
Darbība	Jebkura cita darbība		

Lai norādītu komandu izpildes secību, blokus savā starpā savieno ar līnijām un bultiņām

Uzdevums:

Izveidot vārdiska aprakst algortimu sniegavīra celšanai.

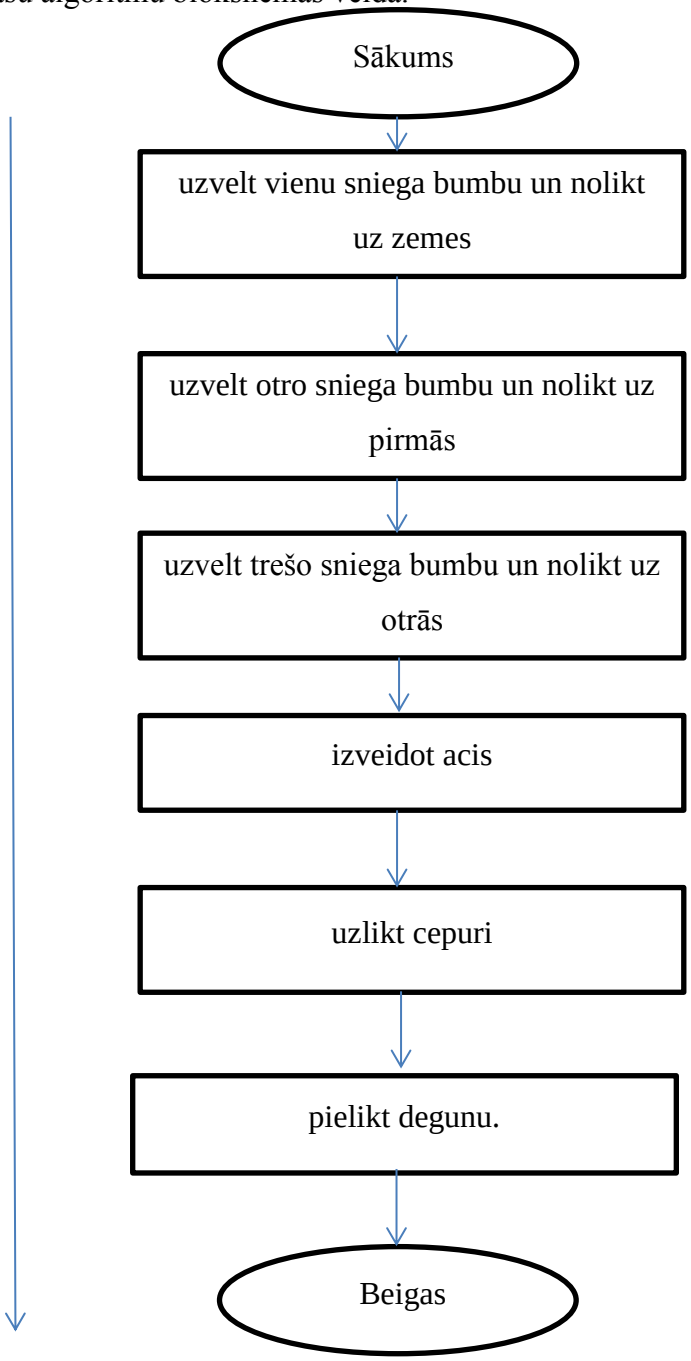
1) uzvelt vienu sniega bumbu un nolikt uz zemes;
2) uzvelt otru sniega bumbu un uzlikt uz pirmās;
3) uzvelt trešo sniega bumbu un uzlikt uz otrās;
4) izveidot acis;
5) uzlikt cepuri;
6) pielikt degunu.

Praktiskais darbs:

Izdomā ikdienišķu darbību, kurai var sastādīt vārdisku algoritmu, uzraksti to.

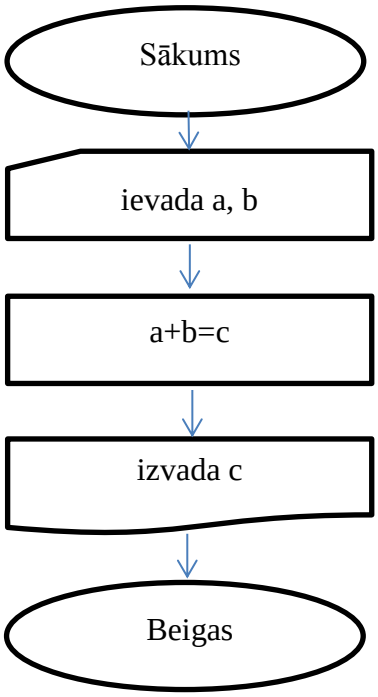
3., 4.nodarbība. Lineāri algoritmi

Iepriekšējā nodarbībā pierakstījām sniegavīra velšanas vārdiskā pieraksta algoritmu. Uzzīmēt šo pašu algoritmu blokshēmas veidā.

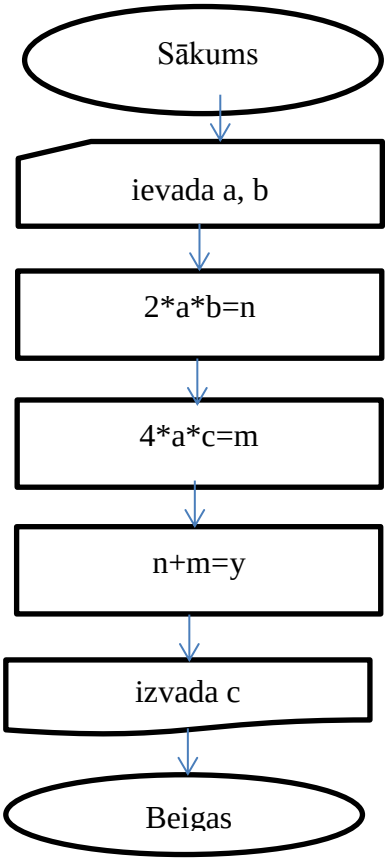


Darbības ir secīgas. Algoritmus, kuru komandas tiek izpildītas tādā secībā, kādā tās pierakstītas, sauc par **lineāriem** algoritmiem.

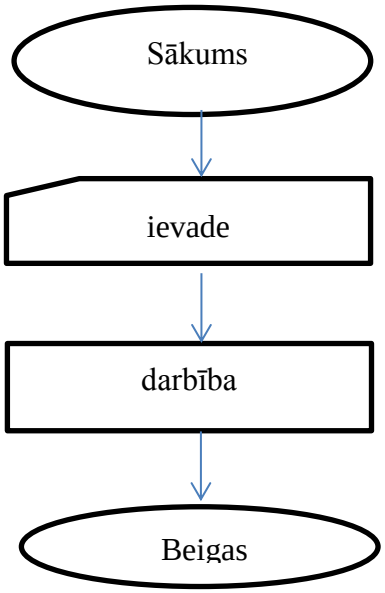
Uzdevums. Uzrakstīt algoritmu bokshēmas veidā matemātiskai izteiksmei a+b=c



Uzdevums: Uzrakstīt algoritmu bokshēmas veidā matemātiskai izteiksmei 2ab+4ac=y
Ieviešam palīgmainīgos n un m: 2ab=n; 4ac=m, līdz ar to izteiksme ir n+m=y



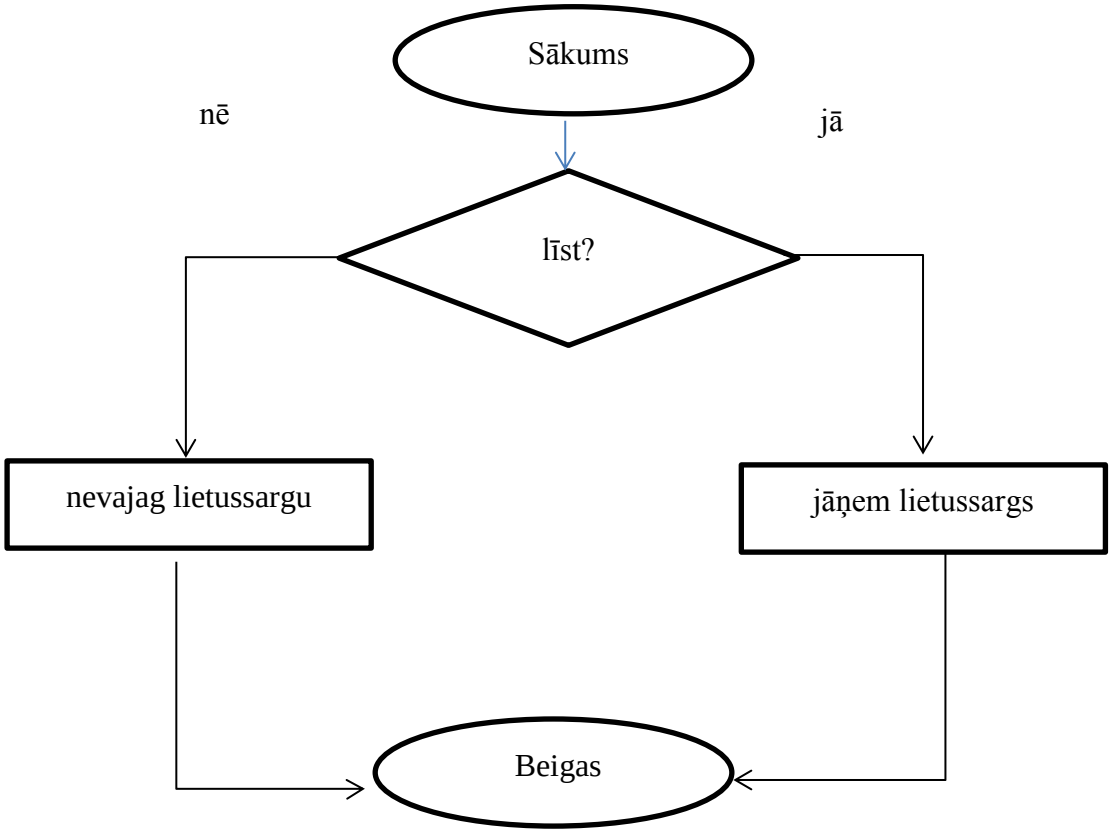
Secinājumi:
lineāra algoritma uzbūve:



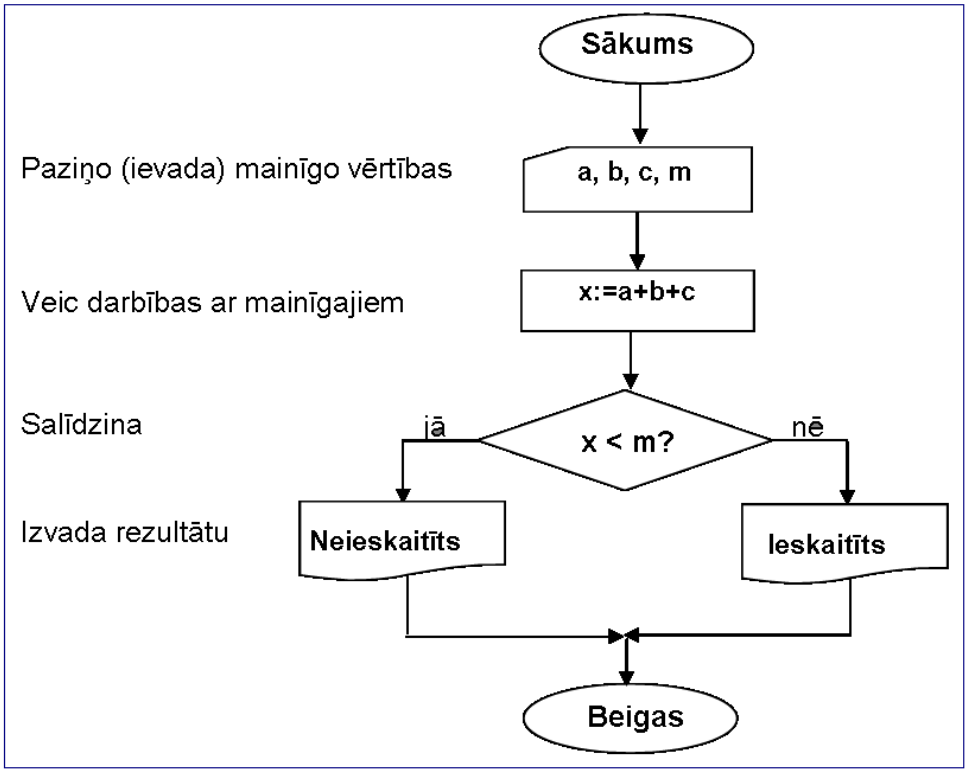
Praktiskais darbs: sastādīt lineāru algoritmu sadzīves situācijai, pamatot to.

5., 6.nodarbība. Sazaroti algoritmi

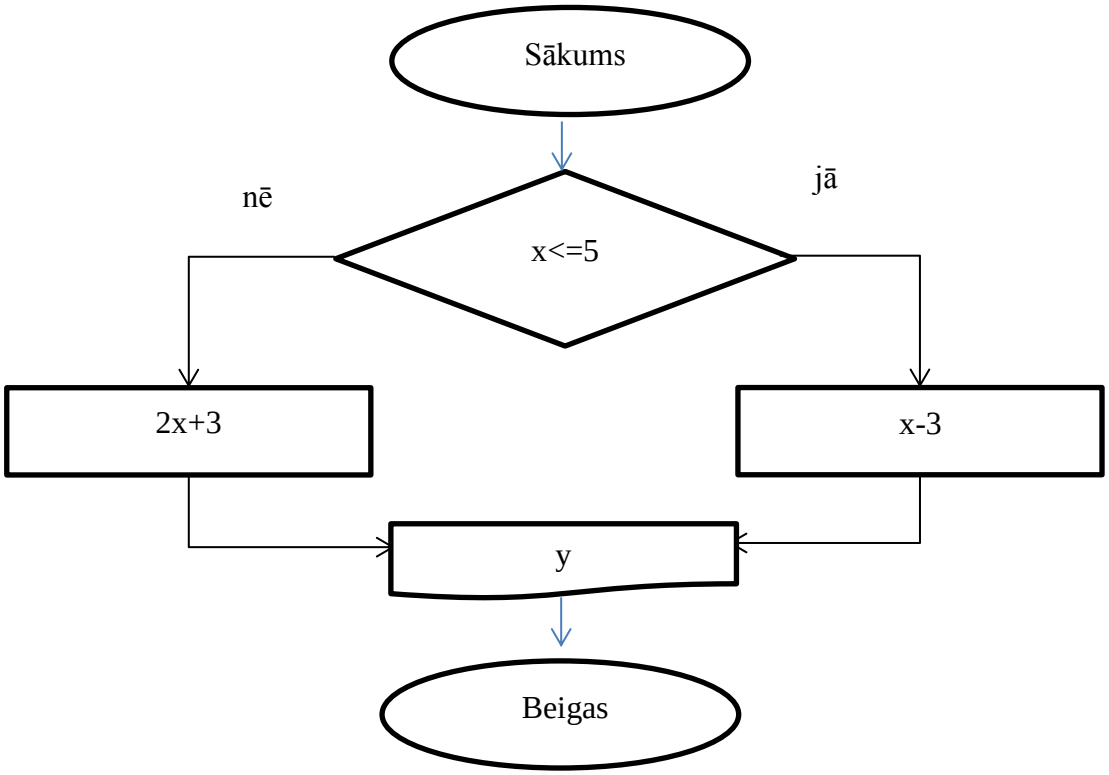
Sazarota strūktura veidojas tad, ja algoritma komandu izpildes secība ir atkarīga no noteikta nosacījuma izpildes.



Sazarota algoritma struktūra

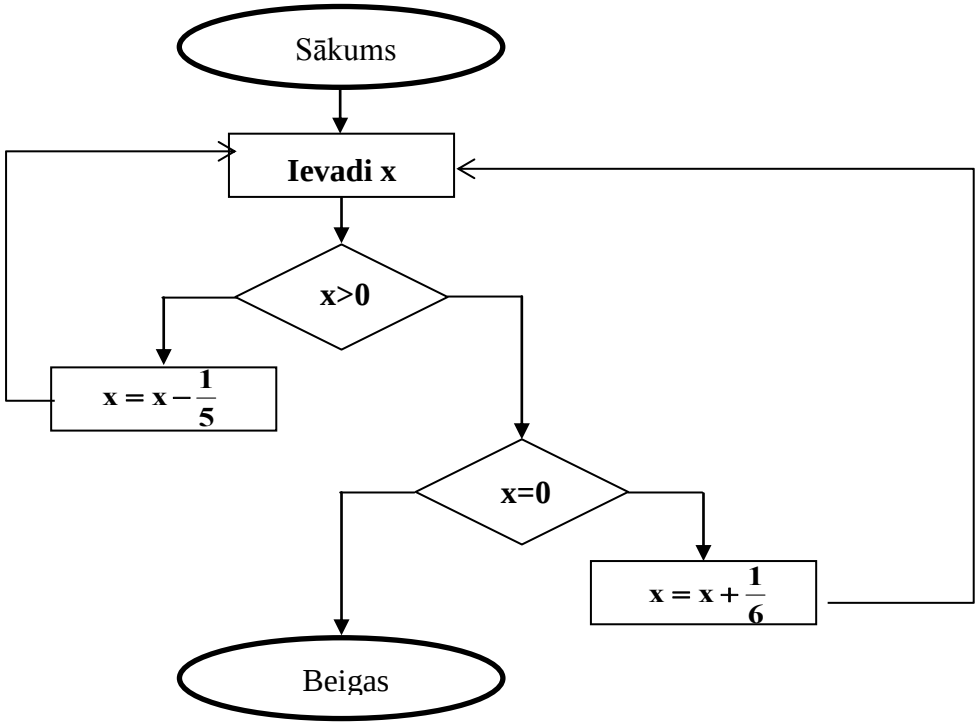


Uzdevums. Sastādīt blokshēmu. Ja x ir mazāks vai vienāds ar 5, y vērtību aprēķina pēc formulas: y=2x+3, pretējā gadījumā y=x-3. Izvada y vērtību. Sākumā jāpiešķir mainīgajam vērtību.



Uzdevums. Atrisināt uzdevumu, izmantojot uzzīmēto blokshēmu. Pierakstīt visus starprezulātus!

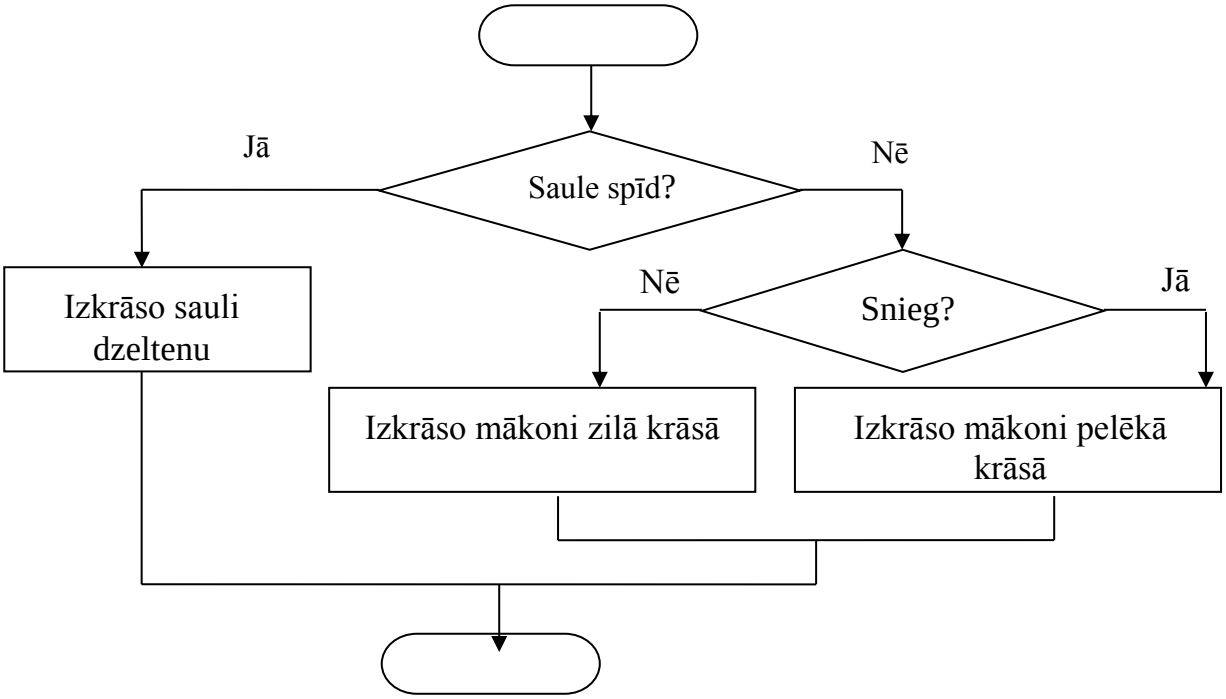
$$x=\frac{1}{15};\quad x=\frac{1}{10};\quad x=-\frac{1}{15};\quad x=-\frac{1}{30};\quad x=1\frac{1}{6}$$



Atbildes

$x = \frac{1}{15}$	$\mathbf{x = \frac{1}{10}}$	$\mathbf{x = -\frac{1}{15}}$	$\mathbf{x = -\frac{1}{30}}$	$\mathbf{x = 1\frac{1}{6}}$
$\mathbf{-\frac{2}{15}}$	$\mathbf{-\frac{1}{10}}$	$\mathbf{\frac{1}{10}}$	$\mathbf{\frac{2}{15}}$	$\mathbf{\frac{29}{30}}$
$\mathbf{\frac{1}{30}}$	$\mathbf{\frac{1}{15}}$	$\mathbf{-\frac{1}{10}}$	$\mathbf{-\frac{1}{15}}$	$\mathbf{\frac{23}{30}}$
$\mathbf{-\frac{1}{6}}$	$\mathbf{-\frac{2}{15}}$	$\mathbf{\frac{1}{15}}$	$\mathbf{\frac{1}{10}}$	$\mathbf{\frac{17}{30}}$
0	$\mathbf{\frac{1}{30}}$	$\mathbf{-\frac{2}{15}}$	$\mathbf{-\frac{1}{10}}$	$\mathbf{\frac{11}{30}}$
	$\mathbf{-\frac{1}{6}}$	$\mathbf{\frac{1}{30}}$	$\mathbf{\frac{1}{15}}$	$\mathbf{\frac{1}{6}}$
	0	$\mathbf{-\frac{1}{6}}$	$\mathbf{-\frac{2}{15}}$	$\mathbf{-\frac{1}{30}}$
		0	$\mathbf{\frac{1}{30}}$	$\mathbf{\frac{2}{15}}$
			$\mathbf{-\frac{1}{6}}$	$\mathbf{-\frac{1}{15}}$
			0	$\mathbf{\frac{1}{10}}$
				$\mathbf{-\frac{1}{10}}$
				$\mathbf{\frac{1}{15}}$
				$\mathbf{-\frac{2}{15}}$
				$\mathbf{\frac{1}{30}}$
				$\mathbf{-\frac{1}{6}}$
				0

Praktiskais darbs. Atvērt sagatavi algoritmi.docx
Ieraksti shēmā izlaistos vārdus. Izpildi algoritmā doto uzdevumu.

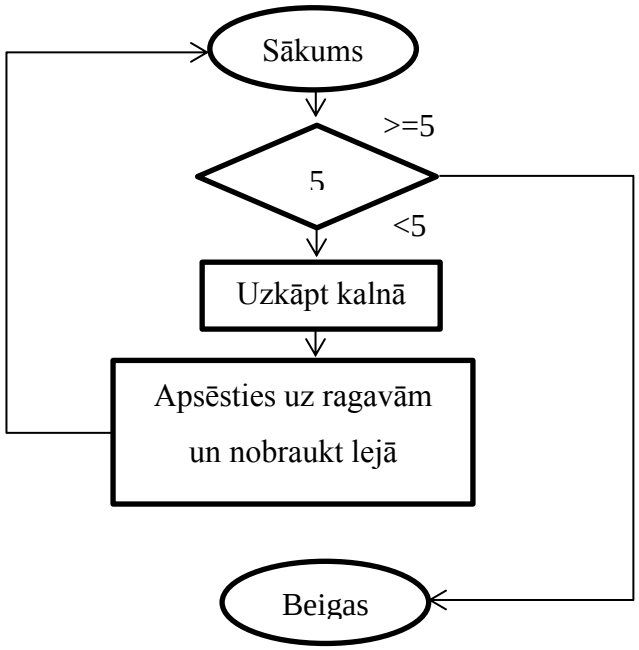


6., 7.nodarbība. Cikliski algoritmi

Cikliska struktūra veidojas tad, ja algoritms satur fragmentu, kas jāatkārto vairākas reizes. Ciklus iedala:

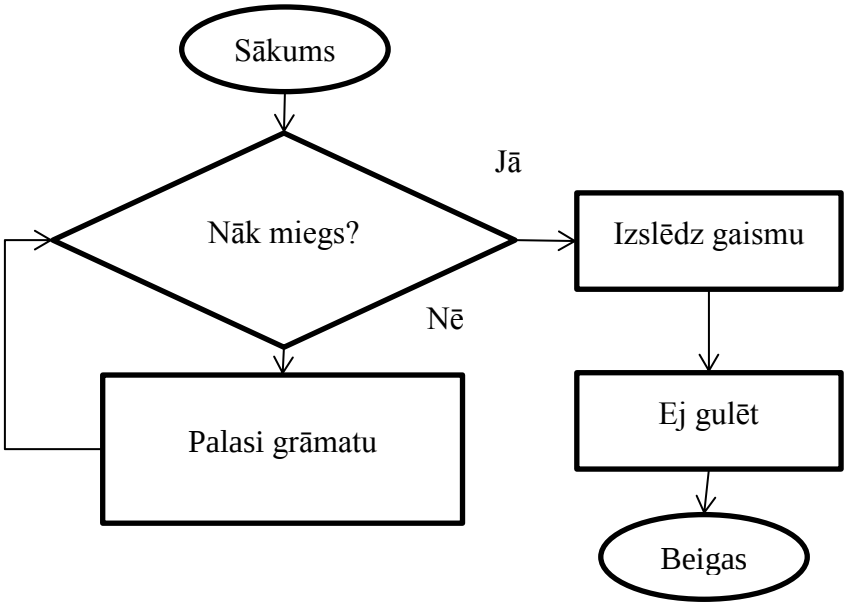
- cikli, kuros komandu fragmenta atkārtošanas reižu skaits ir iepriekš zināms;
- cikli, kuros komandu fragmenta atkārtošanas reižu skaits ir atkarīgs no nosacījuma.

Uzdevums 1.Sastādīt algorimtu: 5 reizes nobraukt no kalna ar ragaviņām.



Uzdevumā dots sākuma nosacījums, ka ciklam jāizpildās 5 reizes.

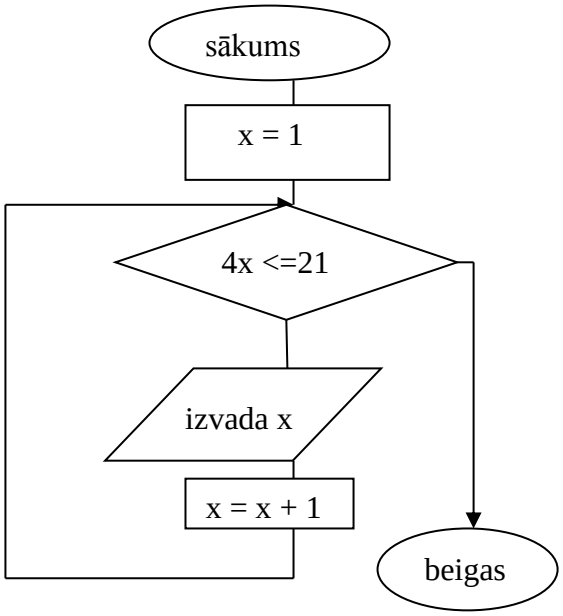
Uzdevums 2. Sastādīt algoritmu: lasi, līdz nāk miegs, tad jāiet gulēt.



Cikls izpildās tik reižu, līdz izpildās nosacījums.

Praktiskais darbs. Ar pakāpenisko pārbaudi atrast dotās nevienādības visus veselos pozitīvos atrisinājumus. Sastādīt blokshēmu un uzrakstīt algoritma izpildes rezultātā iegūtās x vērtības.4 * x <= 21.

Paredzamais rezultāts.
Algoritma izpildes rezultātā
iegūst **x** vērtības:
x = 1, 2, 3, 4, 5.



9., 10.nodarbība. Uzdevumu par algoritmiem risināšana

1.uzdevums. Uzrakstīt vārdisko algoritmu: kā pareizi pārietu ielu pie luksofora.

2.uzdevums. Uzrakstīt dotajam vārdiskajam algoritmam atbilstošu matemātisko izteiksmi.

dots skaitlis x;
pareizināt x ar 5;
atņemt 6;
iegūto skaitli izdalīt ar 4.

3.uzdevums. Sastādīt algoritma blokshēmas veidā: $y = 2a + b$.

4.uzdevums. Sastādīt blokshēmu algoritmam, kas sastāda naturālo skaitļu no 1 līdz 5 reizināšanas tabulu ar 4.

5.uzdevums. Sastādīt algoritmu, kas raksturo pankūku cepšanas darbības.

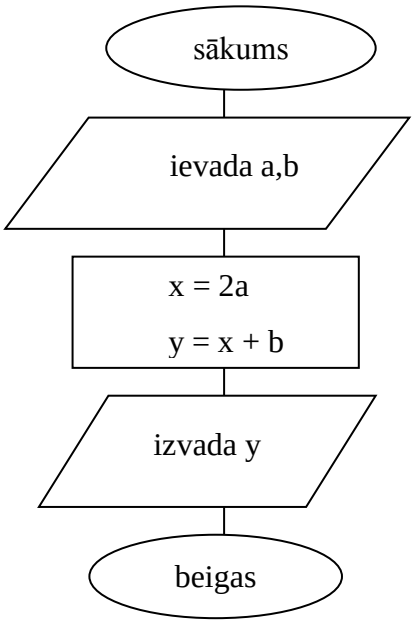
6.uzdevums. Sastādīt trīs uzdevumus, kuru risinājums būs lineārs, sazarots un ciklisks algoritms. Apraksti, ko algoritmā nozīmē katra darbība.

Atbildes:

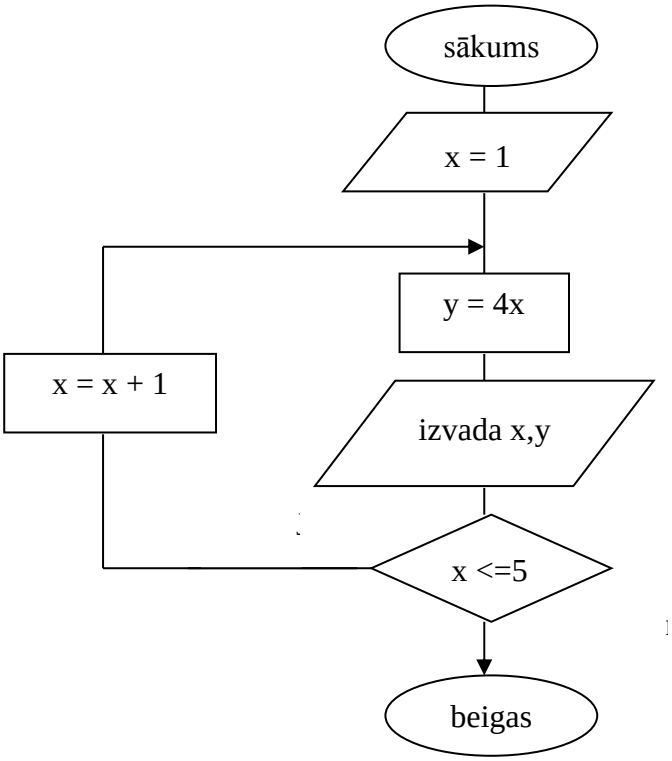
2.uzdevums.

$y = (5x – 6) : 4$

3.uzdevums.



4.uzdevums.



Programmēšanas valodas *Scratch* komandrīku apraksts.

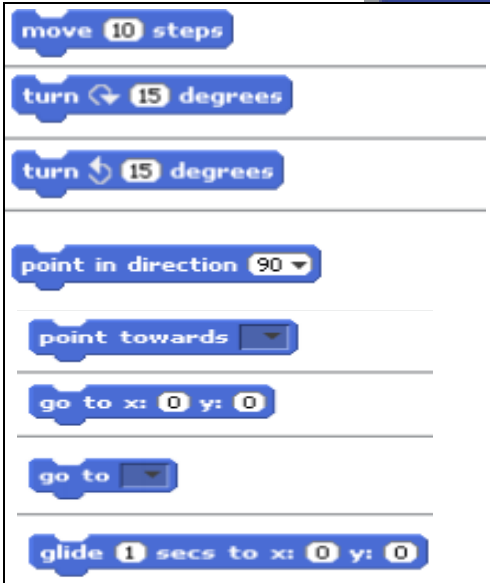
Programmas logs sastāv no 3 daļām. 1.daļas augšpusē ir 8 pogas: kustība, izskats, skaņa, zīmulis, kontroļi, sensitīvi, operātori, mainīgie.

	**Motion** - kustība **Looks** - izskats **Sound** - skaņa **Pen** -zīmēšana		**Control**- kontroļi **Sensing** –sensitivie **Numbers**- skaitļi **Variables**- mainīgie		Palaist skriptu izpildi Pārtraukt skriptu izpildi

Kontroļi.

	Darbojas,kad uzklikšķina uz zaļo karogu. Darbojas, kad nospiež norādīto taustiņu. Darbojas, kad uz uzklikšķina Sprite. Gaida noteiktu sekunžu skaitu, pēc tam turpina dabību ar nākamo bloku. Cikls, nepārtraukti darbojas bloki kuri ir iekšā ciklā. Cikls, atkārto darbības noteiktu skaitu.

Kustība

	Objekts pārvietojas priekšu vai atpakaļ. Rotē pulksteņrādītāja virzienā. Rotē pretēji pulksteņa rādītāja virzienam. Rotē noteiktajā virzienā. Reaģē uz peles kursoru vai citu Sprite. Pārceļ Sprite uz noteiktu x un y pozīciju Pārvieto Sprite ar peles kursoru vai citu Sprite. Pārceļ Sprite vienmērīgi uz norādīto pozīciju.

Izskats

Looks

switch to costume costume1	Izmaina Sprite izskatu, pārejot uz dažādiem Kostīmiem.
next costume	Izmaina Sprite kostīmu uz nākamo kostīmu sarakstā.
costume #	Ziņo sprites kostīma numuru.
switch to background background1	Pāreja uz atšķirīgu fonu.
next background	Izmaina nākamo fona sarakstā.
background #	Ziņo fona numuru.
say Hello! for 2 secs	Parāda Sprite runas burbuli konkrētajā laika posmā.
say Hello!	Parāda Sprite runas burbuli.

Skaņa

Sound

play sound meow	Sāk spēlēt skaņu, kas izvēlēta no izvēlnes, un nekavējoties dodas uz nākamo bloku pat ja skaņa joprojām spēlē.
play sound meow until done	Atskaņo skaņu un gaida.
stop all sounds	Beidz spēlēt visas skaņas.
play drum 48 for 0.2 beats	Spēlē bungas skaņu, no nolaižamās izvēlnes var atlasīt noteiktu sitienu skaits.
play note 60 for 0.5 beats C (60)	Spēlē mūzikas skaņu noteiktu skaitu sitienu.
rest for 0.2 beats	Balstas par noteiktu skaitu sitienu.

sensitīvie

Sensing

touching ?	Ziņo, ja pie Sprite pieskaras norādītais Sprite vai peles rādītājs.
touching color ?	Ziņo, ja uz Sprite pieskaras noteikta krāsa.
color is touching ?	Ziņo Ja pirmā krāsas saskaras ar otro krāsu

Zīmēšana Pen

	Notīra.
	Nolaiž zīmuli, tas zīmēs
	Paceļ zīmuli - nezīmēs.
	Iestata pildspalvas krāsu, pamatojoties uz izvēli no Color Picker.
	Maina zīmuļa krāsu.
	Iestata zīmuļa krāsu uz norādīto vērtību.
	Maina zīmuļa toni.

Skaitļi Numbers

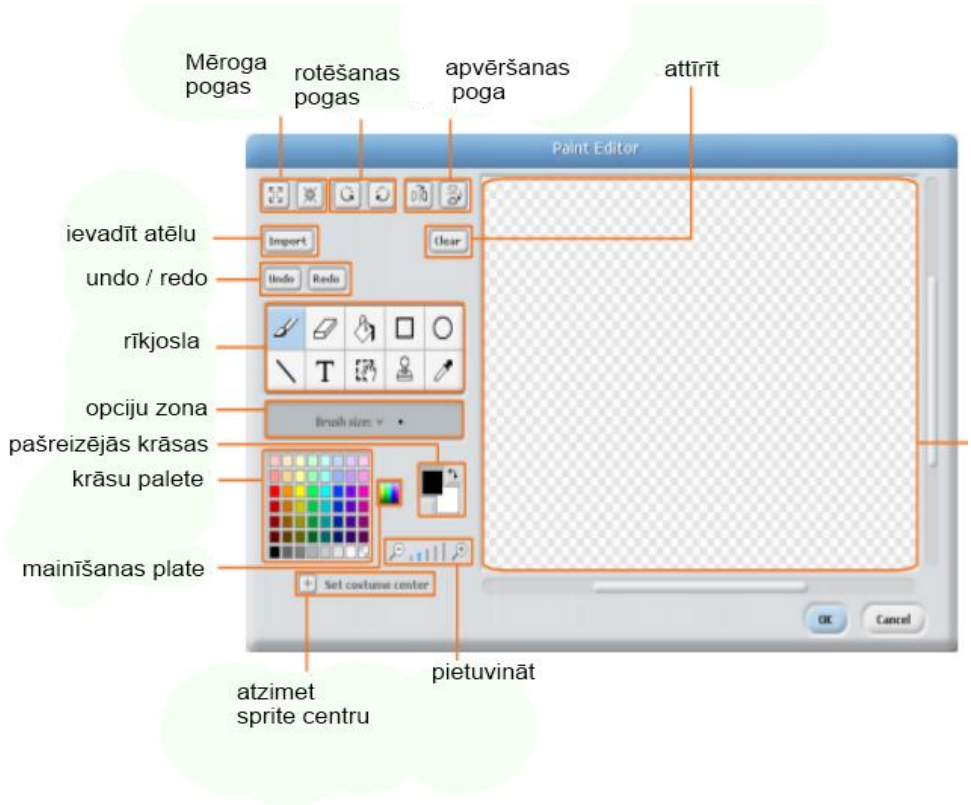
	Saskaita divus skaitļus.
	Atņem otro skaitli no pirmā
	Reizina divus skaitļus.
	Dala pirmo skaitli ar otro.
	Ziņo, ja pirmā vērtība ir mazāka nekā otrā.
	Ziņo, ja abas vērtības ir vienādas.
	Ziņo, ja pirmā vērtība ir lielāka nekā otrā.

Mainīgie

Variables

Make a variable	Noklikšķiniet, lai izveidotu un nosauktu jaunu mainīgo. Kad mainīgais izveidots parādās mainīgā bloki.
Delete a variable	Dzēš visus blokus saistītus ar mainīgo.
score	Mainīgā vērtība.
change score by 1	Maina mainīgo par norādīto vērtību. Ja vairāk nekā viens mainīgais, izmantot nolaižamo izvēlni.
set score to 0	Nosaka mainīgo norādīto vērtību.
show variable score	Parāda mainīgo.
hide variable score	Slēpj mainīgo.

Zīmēšanas logs



1.nodarbība.

Stundas mērķis: iepazīties ar programmas *Scrath* darba vidi un rīkiem.

Aktivizēt programmu *Scratch*. Iepazīties ar *Scratch* darba vidi. Programmai var iestatīt valodu, bet latviešu valodā programma nav pieejama.

Šajā stundā iepazīsimies ar pogām Kontroļi un Kustība.

Loga 2.daļā var iestatīt objekta īpašības, var ierakstīt programmu, ar kuru vadīt objektu. Objektus no *Scratch* bibliotēkas apzīmē kā sprait, atšķiras ar kārtas numuru.

Ja nospiesta poga Scripts parāda programmas tekstu, var ievadīt, rediģēt programmas

tekstu, vadīt objektu. Ar pogu Kostīmi un Skaņa palīdzību var objektam mainīt izskatu, pievienot skaņas efektus.

Ja mēs uz kaķa piespiežam peles kreiso pogu un velkam pa lapu, mainās objekta x, y koordinātes. Poga loga labajā stūrī palaiž un aptur programmas izpildi.

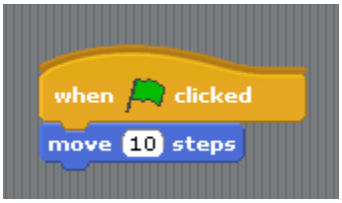
Pogas  pārslēdz programmas skatus.

Pirmā programma.

Kustība uz priekšu 10 soļus.

Vispirms likt kontroli, kas norāda, ka uzspiežot uz karodziņa tiks aktivizēta programma, tad kustība 10 soļi. Pārbaudi kā programma izpildās. Kaķis pārvietojas uz priekšu.

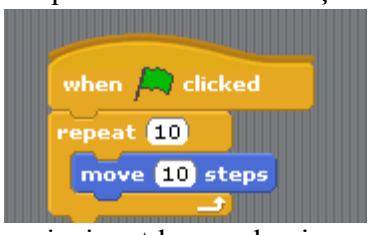
Pievērs uzmanību kā komandas savienojas, līdzīgi kā Lego klucīši. Ja komandas



nesavienojas, tātad, iespējams tiek veidots kļūdains programmas teksts.

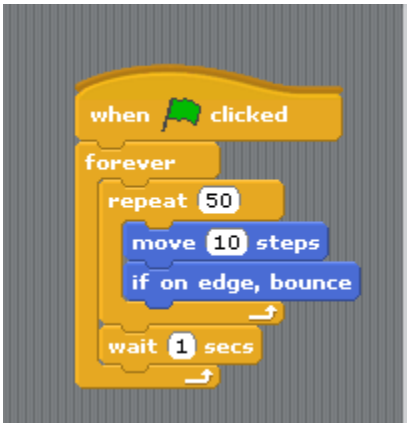
Lai kaķis pārvietotos tālāk, pievienot kontroli „atkārtot", norādot cik reizes. Testēt programmu. Kaķis strauji pārvietojas, bet tā neizskatās pēc iešanas.

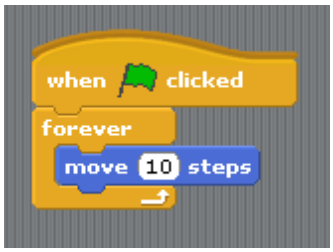
Noņemt ciklu „atkārtot" un ielikt ciklu „nepārtraukti", testēt kā strādā. Šis cikls nodrošina nepārtrauktu darbību, līdz kaut kas nepārtrauc darbību. Kaķis aiziet līdz lapas malai



un tur paliek. Lai izmainītu kustības virzienu, pievienot komandu „ja mala, atgrūzties". Kaķis staigā no vienas malas līdz otrai.

Patstāvīgais darbs.
Izveidot programmu, izksaidrot ko programmā nozīmē katra komanda.



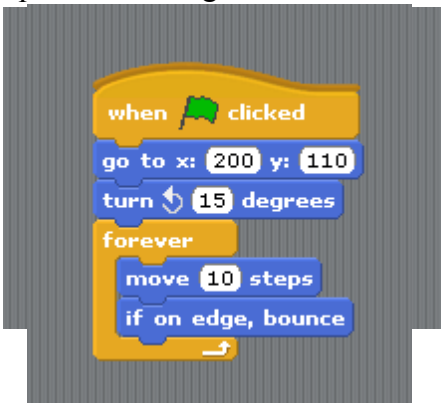


Jautājumi patstāvīgajam darbam:
1) Cik ciklu ir programmā? Nosauciet tos.
2) Kurš cikls ir ārējais, kurš iekšējais?
3) Kādas komandas izpilda cikls „atkārtot”?
4) Kādas komandas izpilda cikls „nepārtraukti”?
5) Cik soļu noiet kaķis līdz trešo reizi apstājas uz sekundi.

2. nodarbība. Objektu vadīšana

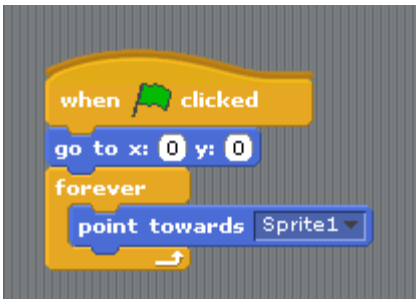
Objektu vadīšanā ir jāpievērš uzmanība objekta (Sprite) koordinātēm. Pēc noklusējuma Sprite (0,0) koordinātes ir lapas vidū. Ievietot otru objektu var ar komandu *Import*. Pāriet atpakaļ uz spraitu Kaķis. Izveidot programmu – kaķis kustās pa noteiktām koordinātēm, cikls nodrošina nepārtrauktu darbību.

Pāriet uz spraitu Pele. Izveidot programmu peles kustībām. Pele kustās pa noteiktām koordinātēm. Pagriežas pa noteiktiem grādiem. Pārbaudīt kā objekti pārvietojas.



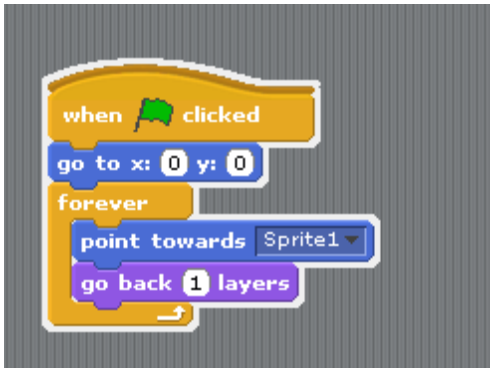
Ievietot trešo spraitu Lauvu. Izveidot programmu: Lauva stāv uz (0,0) koordinātes un pagriežas kaķa virzienā, kad tas tuvojas.

Komanda *point towards* liek objektam griezties izvēlētā objekta virzienā. Cikls *forever*



nodrošina objekta kustības atkārtošanos. Nelietojot ciklu, darbība notiktu tikai vienu reizi. Noskatieties izveidoto animāciju.

Pievērsiet uzmanību – objekti pārvietojoties pārklājas, tas nozīmē, ka katrs objekts pārvietojas savā slānī. Lai pārvietotu uz citu slāni var izmantot komandu *go back ... layers*, vai komandu *go to front*. Izdarīt izmaiņas programmas tekstā un noskatīties animāciju. Salīdzināt izmaiņas.



Lai visi objekti ietilptu logā, objektus var samazināt un palielināt pēc vajadzības ap pogām loga augšpusē. Pirmā poga dublē objektu, turklāt kopā ar programmas tekstu. Otrā poga dzēš objektu. Trešā poga palielina, ceturtā- samazina objektu. Nospiežot kādu no pogām, kursora izskats mainās. Objekts mainās, ja kursors tiek novietots uz objekta un tiek noklikšināts.

Objekta izmērus var mainīt ar komandām 

. Ar komandas *change size by* var mainīt objekta izmērus pēc norādītiem punktiem. Komanda *set size to ... %* mainīs objekta izmērus procentuāli atbilstoši oriģinālam. Oriģināla izmērs ir 100%, tātad, lai samazinātu divas reizes, lauka jāraksta 50%, ja grib divas reizes palielināt, laukā jāraksta 200%

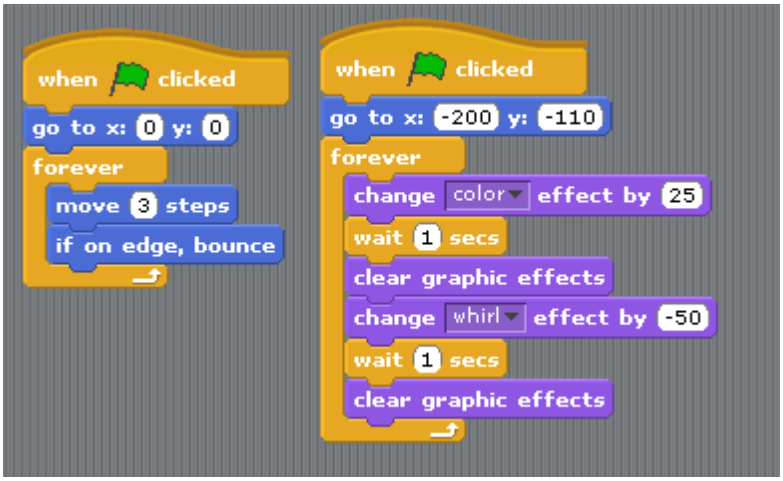
Patstāvīgais darbs.

1) izveidot jaunu projektu „Zirgs”.
2) ievietot divus objektus, zirgs (no mapes *Animals*) un klints (no mapes *Things*);
3) izveidot programmu - lai zirgs pārvietojas no lapas vienas malas līdz otrai pa priekšu kalnam;
4) mainīt programmas tekstu: zirgs kustās no vienas malas līdz otrai aiz kalna.
5) Saglabāt projektu.

3. nodarbība. Vienlaicīga un secīga darbību izpilde

Iepriekšējā nodarbībā izveidojāt algoritmu, kurā visi spraiti uzsāka darbību vienlaicīgi. Šajā nodarbībā rakstīsim vienam objektam vairākus skriptus, kurus objekts izpildīs vienlaicīgi. Izmainīt kaķa skriptu: kaķis pārvietojas no vienas malas līdz otrai. Izveidot otru skriptu:

1) kaķis maina krāsu, gaidīt sekundi;
2) grafiskais efekts tiek noņemts;
3) kaķim tiek mainīta āriene (efekts *whirl*), gaidīt sekundi;
4) grafiskais efekts noņemts.

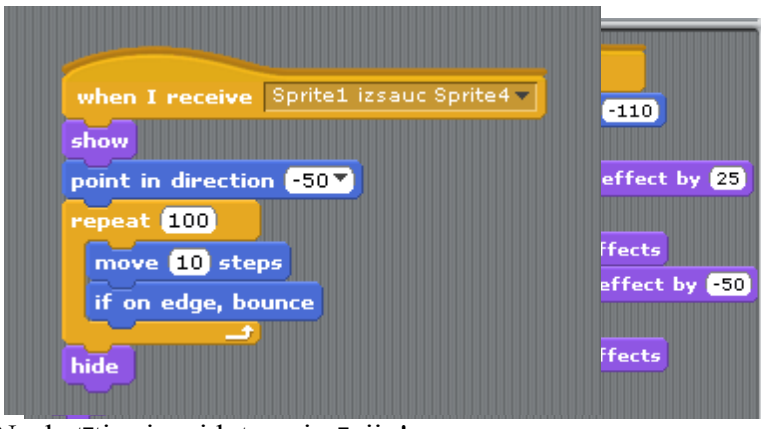


Violētais komandu bloks attiecas uz objekta ārienes maiņu: krāsu, izmēru, u.c.

Objekti animācijā darbību uzsāk vienlaicīgi. Kā izdarīt, lai tie sāktu darbību katrs savā laikā? To var panākt ar kamandu *wait*, bet tad reāli figūras darbību sāk vienmaicīgi, tikai otra nogaida. Lai izveidotu secīgu darbību izpildi, pirmais objekts pabeidzis darbību dod zīmi otram, savukārt otrs objekts to uztver. *Scratch* izmanto komandu *broadcast* un *when I receive.* Komandā *broadcast* jāieraksta teksts, piem. „Sprite1 izauc Sprite2", šis teksts parādīsies komantā *when I receive* un izpildīsies. Nedrīkst aizmirst komandas *show* un *hide,* kuras nodrošinā lai objekts parādītos un pazustu no ekrāna.

Izmainītais progammas teksts kaķim:

Izmainītais programmas teksts taurenim.



Noskatīties izveidoto animāciju!

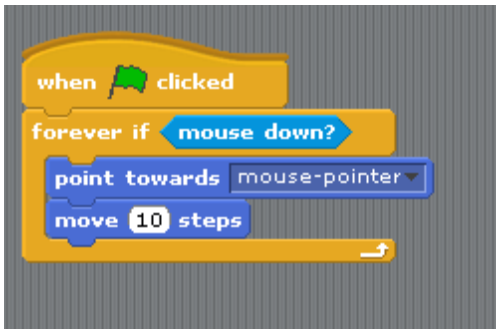
Patstāvīgais darbs.

1) Pievienot trešo objektu.
2) Objekts parādās un pārvietojas pa ekrānu pēc otrā objekta.
3) Jaunajam objektam uzlikt efektus: vispirms lēnām palielinās, tad lēnām samazinās, lēnām atgūst iepriekšējos izmērus.

4. nodarbība. Nosacījumi un mainīgie, vides interaktivitāte

Šajā nodarbībā mācīsimies izveidoto animāciju padarīt interaktīvu. Ar jēdzienu „interaktīvs" jāsaprot dažādas vides mijiedarbību. Tas nozīmē – objektu vada cilvēks ar klaviatūras taustiņu vai peles palīdzību.

Cikls *forever if* nodrošina to, ka cikls izpildās tikai pie nosacījuma, šajā gadījumā komandas *mouse down?* – kad nospiesta peles poga. Komanda *point towards mouse-pointer*



seko kursoram pa x asi.

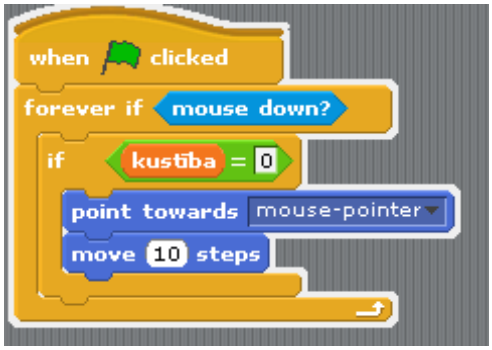
Uzdevums. Ievietot otru objektu, nokopēt pirmā objekta programmas tekstu otram objektam. Pārbaudīt, kā strādā programma.

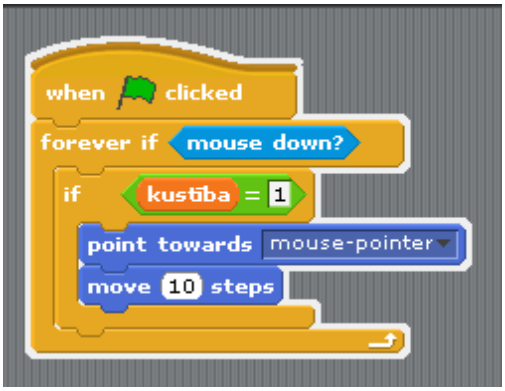
Pārveidot programmu tā, lai objekti seko kursoram pēc kārtas, vispirms viens, tad otrs. Lai pārslēgtos starp objektiem izmantosim taustiņu „atstarpe". Lai programmu pārveidotu jāizmanto *mainīgie*.

Vispirms ir jāizveido mainīgais, šajā gadījumā „kustība", lai varētu nodrošināt pārslēgšanos no viena objekta uz otru. Ja mainīgais „kustība"=0, kustās 1.objekts, ja

mainīgais „kustība"=1, kustās 2.objekts. Mainīgo pārslēgšanos nodrošina klaviatūras taustiņš „atstarpe."

Nosacījums *if-else* nodrošina nosacījumu izpildi: pie *if* tiek nodefinēts nosacījums, kas var būt arī matemātikas operācijas, kas notiek ja neizpildās, ir nodefinēts pie *else*.

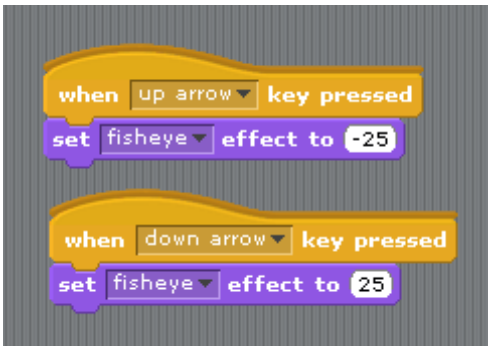




Mainīgais nodefinēts. Izmainīt objektu programmas, tām piesaistīt mainīgos.

Pievienot programmai komandu *if*, kurā ievietojam mainīgo. Vienam objektam nodefinējam „kustība"=0, otrai - „kustība"=1. Rezultātā testēt programmu, ar taustiņu „atstarpe" var pārslēgties starp objektiem, kuri tiek vadīti ar peli.

Iepsējams mainīt objekta izskatu ar taustiņu palīdzību. Izveidot vienkāršas progrmmas.



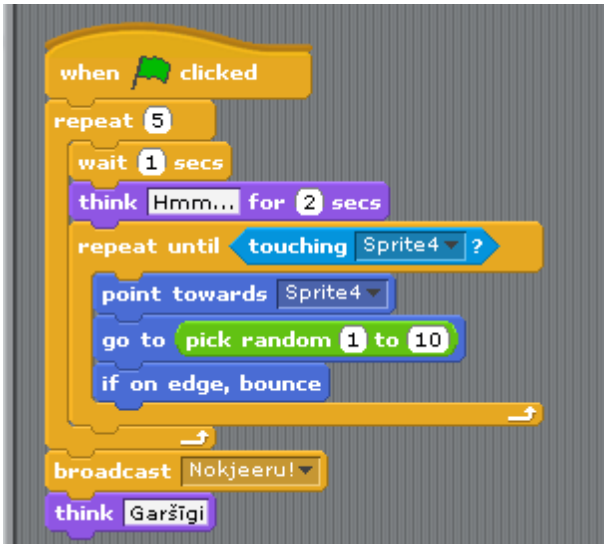
Var izveidot programmas, kas maina objekta izskatu, piemēram palielina un samazina, maina krāsu.

Patstāvīgais darbs.
Sastādīt programmu:
1) lietotājs var vadīt objektu ar kursora vadības taustiņiem,
2) ja tiek piespiests kāds cits klaviatūras taustiņs, tad objekts domā dažādas domas.

5. nodarbība. Gadījuma skaitļi.

Uzdevums: zivtiņa cenšas noķert barību. Sākumā ziviņa sāv un domā, tad pagriežas pret barību un dodas pie tās, līdz pieskarās tai. Tas atkārtojas 5 reizes.

Programmas tekstā komanda *go to* skaitļi brīvi izvēlēti robežās no 1 līdz 10. Šajā gadījumā tas ir ātrums, ar kādu zivs pārvietojas. Zivs 5 reizes padomās „Mmmm" un vienu

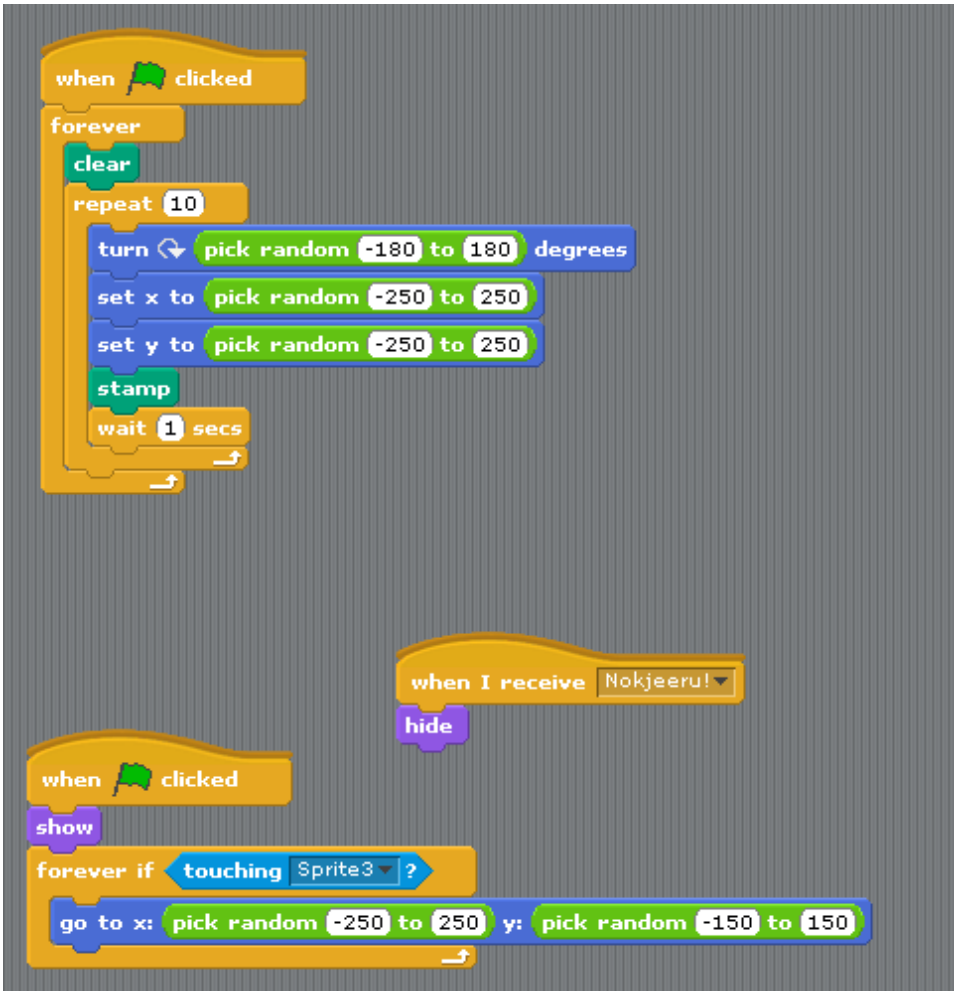


reizi „garšīgi".

Izveidot programmu otram objektam, kas ir ēdiens.

Otrs objekts brīvi pārvietojas pa laukumu. Komanda *stamp* nodrošina objekta dublēšanos, pēc programmas teksta 10 reizes.

Animāciju var izveidot interesantāku: ja mainās fons – mainās zivs izskats.



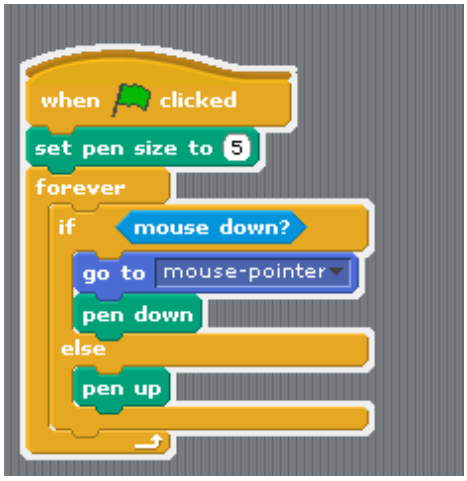
Patstāvīgais darbs.

1) Izdomāt scenāriju, kur ir jāizmando gadījuma skaitļi.
2) Realizēt ro *Scratch* vidē.

6. nodarbība. Zīmēšana ar *Scratch*

Programmēšanas vidē *Scratch* var zīmēt. Zīmēšanai ir komandu grupa *Pen*.

1.uzdevums. Izvēlēties līnijas biezumu – tas ir 5 punkti. Programmas teksts ir cikls

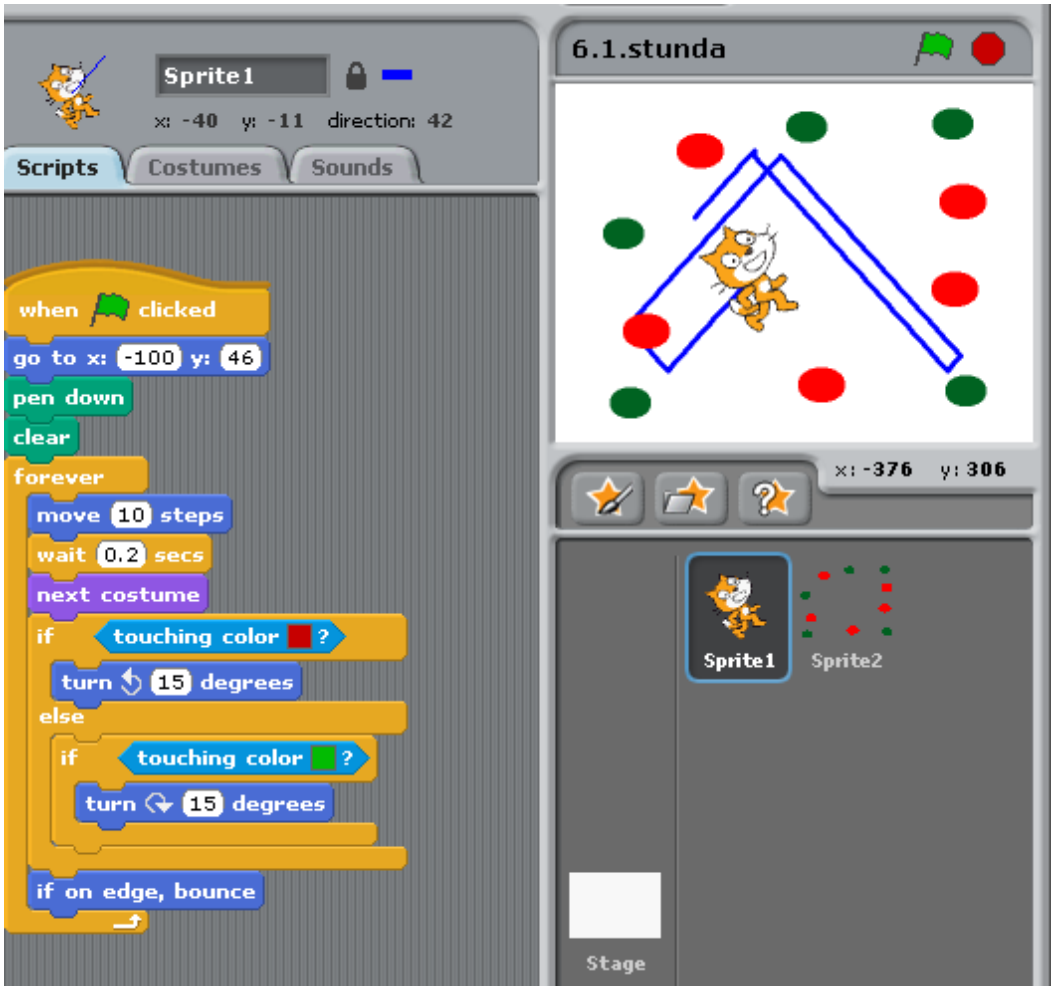


forever, nodrošina nepartrauktu darbību. Ja piespiesta peles kreisā poga, zīmulis nolaists, tas nozīmē, ka zīmē, ja nē, tad pacelts – nezīmē.

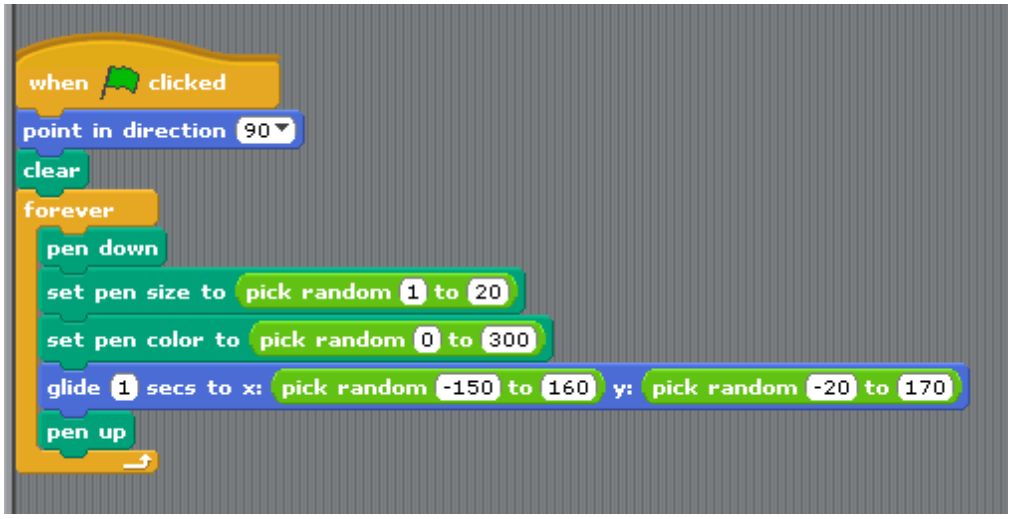
2.uzdevums.

Uzrakstīt programmu: objekts zīmē līniju. Ja saskaras ar sarkano krāsu, maina virzienu, ja saskaras ar zaļo krāsu – pagriežas pretējā virzienā.

3.uzdevums.

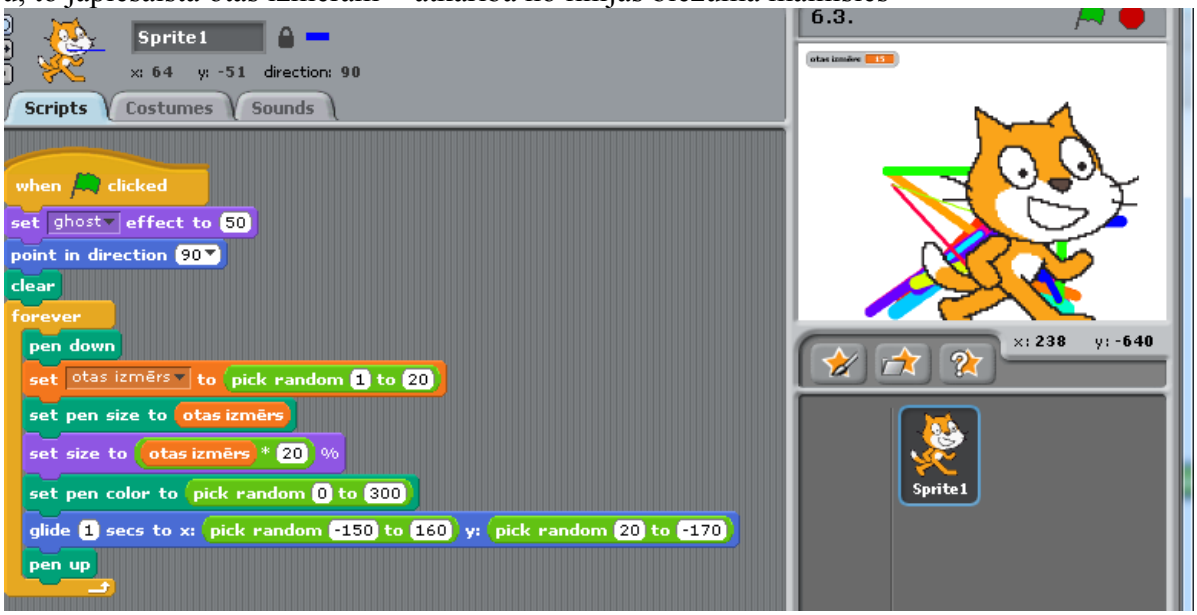


Uzrakstīt programmu, kur objekts zīmē līnijas: brīvi izvēlētas krāsas no 1 līdz 20, brīvi izvēlēta krāsa no 0 līdz 300. Objekts pārvietojas pa brīvi izvēlētām x, y koordinātēm, zīmē ar brīvi izvēlētām krāsām un brīvi izvēlētām līnijām.



Patstāvīgais darbs.
Izveidot programmu:
1) objekts zīmē brīvi izvēlētās krāsās (0;300);
2) brīvi izvēlēts līnijas biezums;
3) otas izmērs (1;20)
4) objekta izmērs mainās par 20% atkarībā no līnijas biezuma;
5) mainīgās x koordinātes (-150;160); y koordinātes (20; -170)

Objekta izmērs ir proporcionāls līnijas biezumam 20%. Objektam jāuzliek efekts ghost. Lai varētu uzrakstitīt programmu, jānodefinē mainīgas, tas ir otas izmērs. Definējot objekta izmēru, to jāpiesaista otas izmēram - atkarībā no līnijas biezuma mainīsies

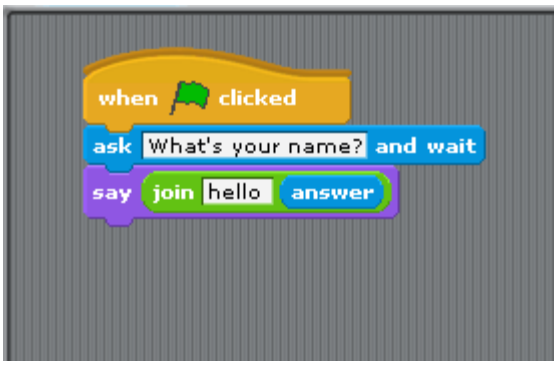


Sasniedzamais rezultāts.

7. nodarbība. Dialogs ar programmu.

Programma *Scratch* nodrošina iespēju veidot lietotājam dialogu ar programmu, šīs komandas atrodas sadaļā *Sensing*.

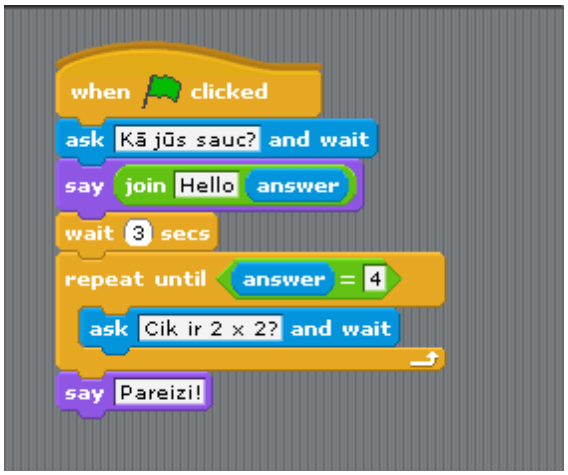
1) uzrakst programmu, kas jautā „Kā jūs sauc”
2) Izveidot atbildes komandu : Hello, ..., kura tiek izvadīta uz ekrāna.



To nodrošinā komanda *ask What`s your name? and wait.* Frāzes *„Kā tevi sauc”* vietā var rakstīt jebkuru tekstu. Kad komanda izpildās, loga apakšējā daļā parādās logs, kurā ievadīt atbildi. Kad atbilde ievadīta, jānospiež *Enter*. Kad tā apstiprināta, atbildes lauks pazūd no ekrāna. Pārbaudīt programmu.

Izveidot programmu:

1) Kaķis jautā „Kā tev sauc?”
2) Kaķis sasveicinās, gaida 3 sekundes.
3) Jautā cik ir 2x2?
4) Ja atbild „4”, izvada tekstu „Pareizi!”



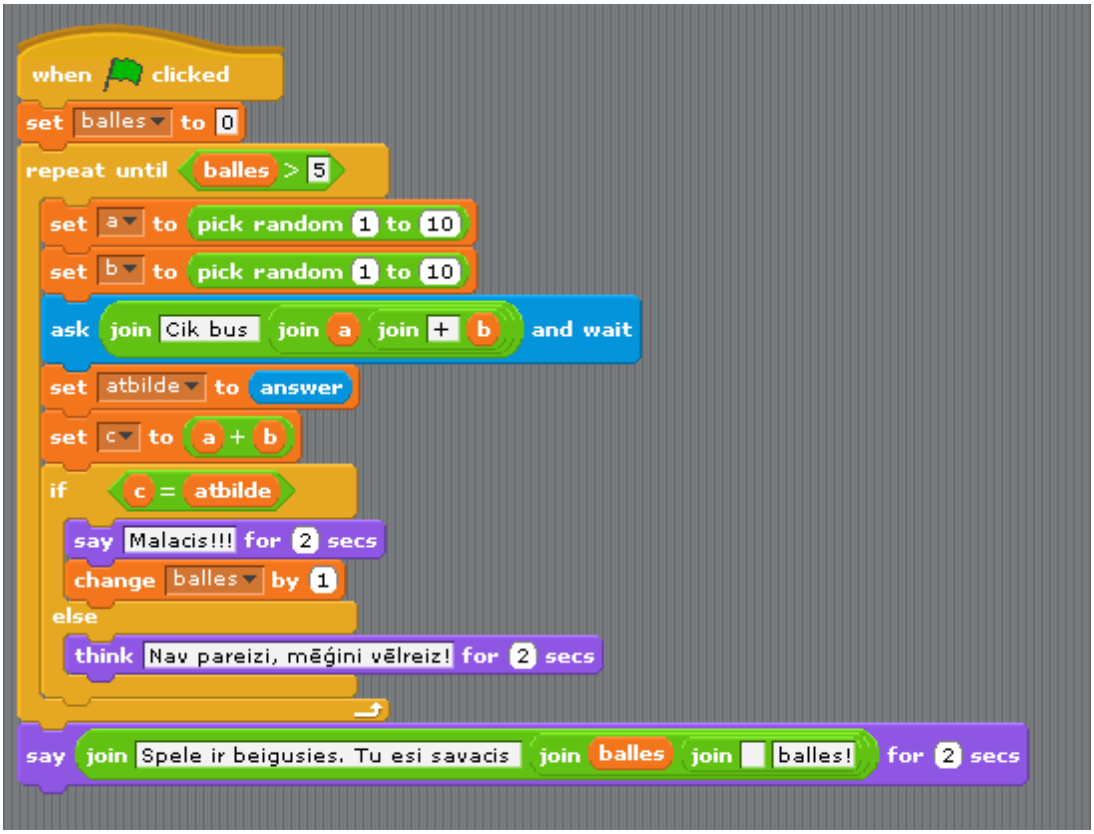
Pārbaudīt izveidoto programmu.

Sarežģījam uzdevumu. Izveidot programmu:

1) kura realizē saskaitīšanas darbības 10 apjomā.
2) nodefiēt mainīgo – balles, a, b, c,atbilde;

3) ja atbilde ir pareiza, seko uzslava „Malacis”;
4) ja atbilde nepareiza – „Nav pareizi, mēģi vēlreiz!”

Vispirms nodefinē iespējās balles, cik var savākt. Ciks atkārtojas, līdz savāktas 5



pareizas atbildes. Komanda *set a to pick random 1 to 10* nodefinē: mainīgie tiks brīvi izvēlēti. Jautājuma blokā ar operātoru palīdzību nodefinēta darbība.

Cikls *if... else..* nodrošina atbildes pārbaudi, pie *if* zara, kas notiek, ja atbilde ir pareiza, pie *else* zara, kas izpildās, ja atbilde ir nepareiza.

Spēles beigās tiek izvadīts iegūtais rezultāts.

Patstāvīgais darbs.

1) Sastādīt programmu, kura jautā lietotājam par cik procentiem palielināt vai samazināt kaķi. Pārbaudīt, vai izpildās.
1) uzrakstīt programmu, kura pieprasa lietotājam kādu figūru zīmēt (cik stūri) Figūra, izpildoties programmai, tiek uzzīmēta.

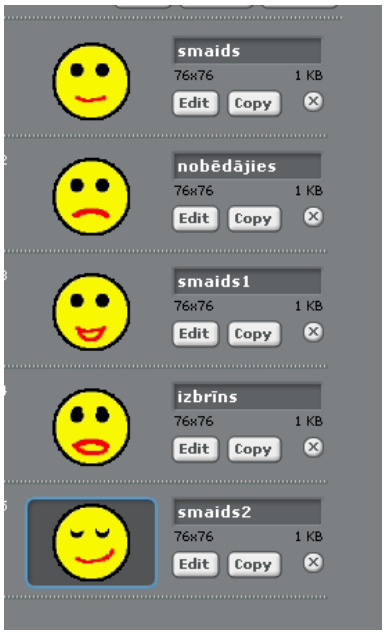
Otrajā uzdevumā ir jānodefinē divi mainīgie.

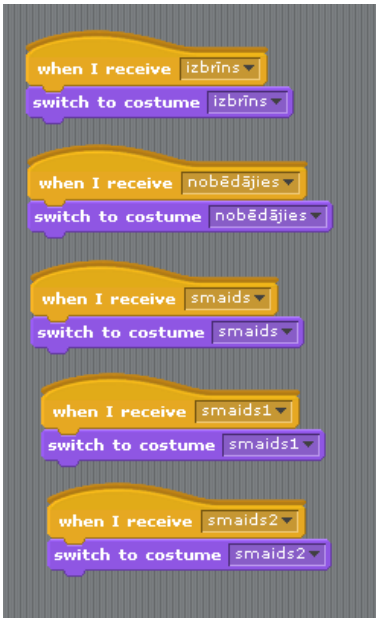
8. nodarbība. Objektu un kostīmu veidošana

Strādājot *Scratch* vidē var lietot objektu bibliotēku, bet var pats veidot zīmējumu. Zīmēšana notiek līdzīgi, kā MS Paint. Šaja vidē var:

- izmainīt objekta izmērus, pagriezt objektu horizontāli, vertikāli;
- importēt gatavu objektu, lai to izmainītu;
- zīmēšanai izmantot dažādus rīkus: otu, līniju, ģeometriskas figūras;
- mainīt līnijas biezumu;
- izvēlēties krāsu, aizpildīt objektu ar izvēlēto krāsu;
- dzēst detaļas;
- strādāt ar tekstu;
- mainīt fona krāsu;
- mainīt objekta izmērus.

Veidot zīmējumu: smaids. Sastādīt skriptu. Ja piespiež konkrētus klaviatūras taustiņu, mainās sejiņa.





Patstāvīgais darbs.

Strādāt zīmēšanas vidē. Ievietot no bibliotēkas objektu, kopēt, izmantojot zīmēšanas rīkus mainīt zīmējumu. Uzrakstīt skriptus, kas nodrošina objektu maiņu tad, ja kursors pieskaras kādai konkrētai krāsai.

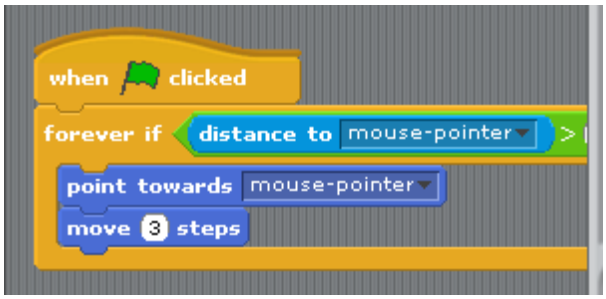
9. nodarbība. Objektu bibliotēkas izmantošana

Strādājot ar *Scratch* izmantot var objektu bibliotēku. Bibliotēku var papildināt, piemēram, varam pievienot iepriekšējā nodarbībā izveidoto smaidu. Lai to paveiktu:

- atvērt projektu, atlasīt objektu;
- izvēlēties komantu *File/Export Sprite;*
- dialoga logā izvēlēties vajadzīgo mapi;
- nosaukt un saglabāt objektu.

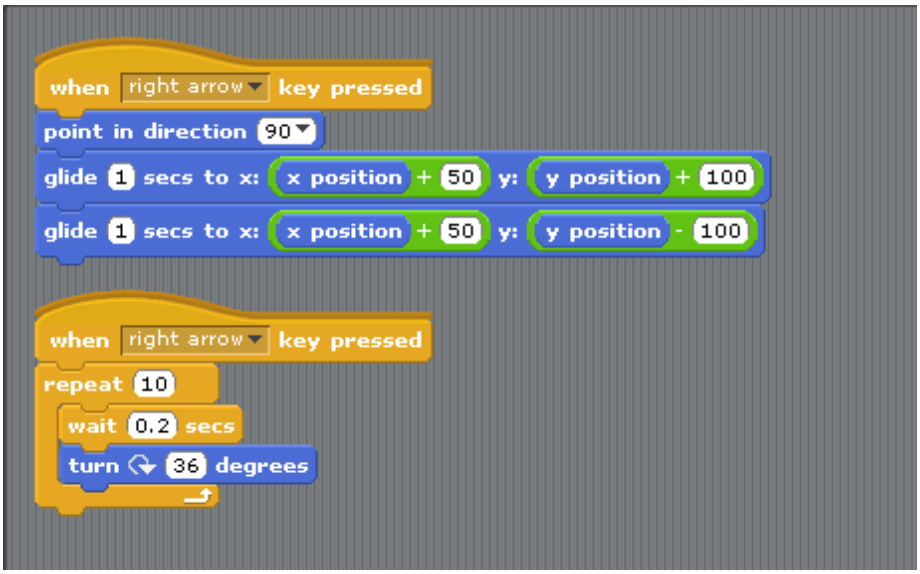
Turpmāk to varēs lietot kā objektu no bibliotēkas. Pētīsim iebūvētās bibliotēkas objektu skriptus. Paņemt objektu *hungry fish.*

Skripta tekstā – zivs seko kursoram, līdz attālums līdz kursoram ir lielāks par 10



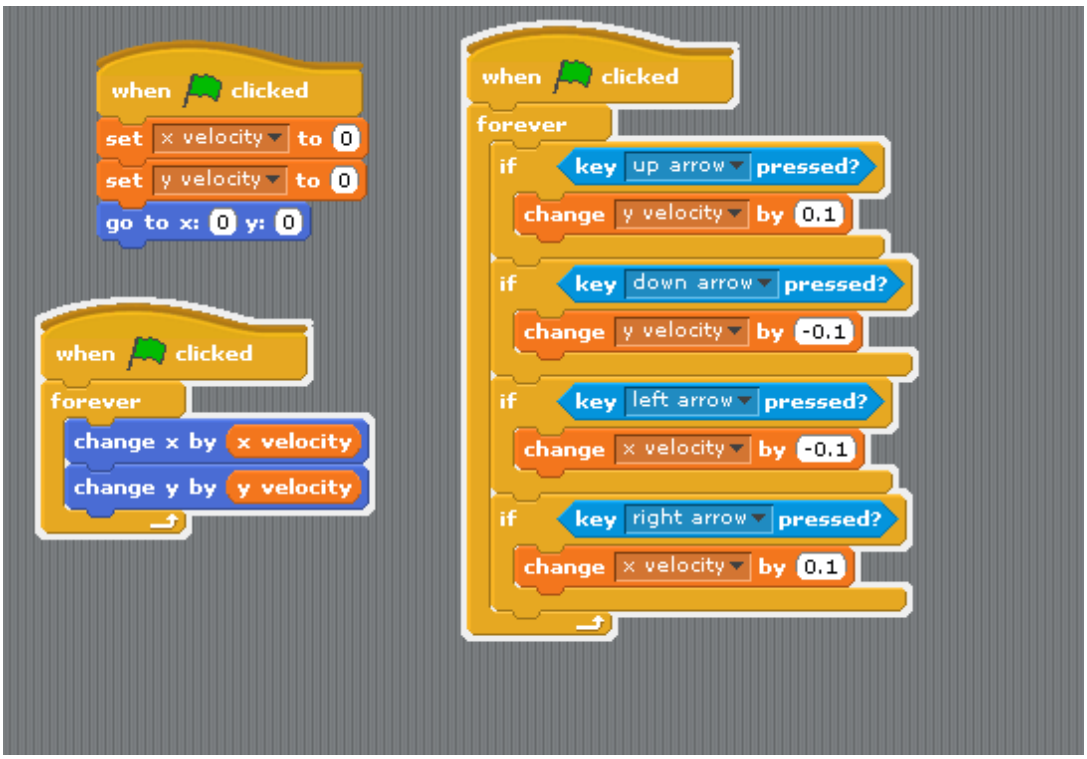
punktiem. Šo attālumu var arī manīt.

Objekts *Jump-flip monster* ir interesanta programma, kurā objekts met kūleni. Objekts met kūleni, kad tiek nospiests taustiņš uz klaviatūras. Pirmā komanda liek objektam pārvietoties uz priekšu pa 50 punktiem un uz augšu 100 punktiem, otrā komanda liek objektam kustēties 50 punktus uz priekšu un 100 uz leju. Komanda *point in direction 90* liek objektam atgriezties stāvoklī, kādā bija pirms kustības uzsākšanas.



Pārbaudīt kā strādā programma.

Izvēlēties objektu *Friction marble.* Izpētam programmas tekstu. Programmas tekstā lietots mainīgais -x velocity un y velocity, kuru vērtība sākumā ir 0. Kad piespiež bultiņtaustiņus, objekts maina virzienu. Ja bultiņtaustiņu tur piespiestu, tad objekts pārvietojas arvien ātrāk un ātrāk.



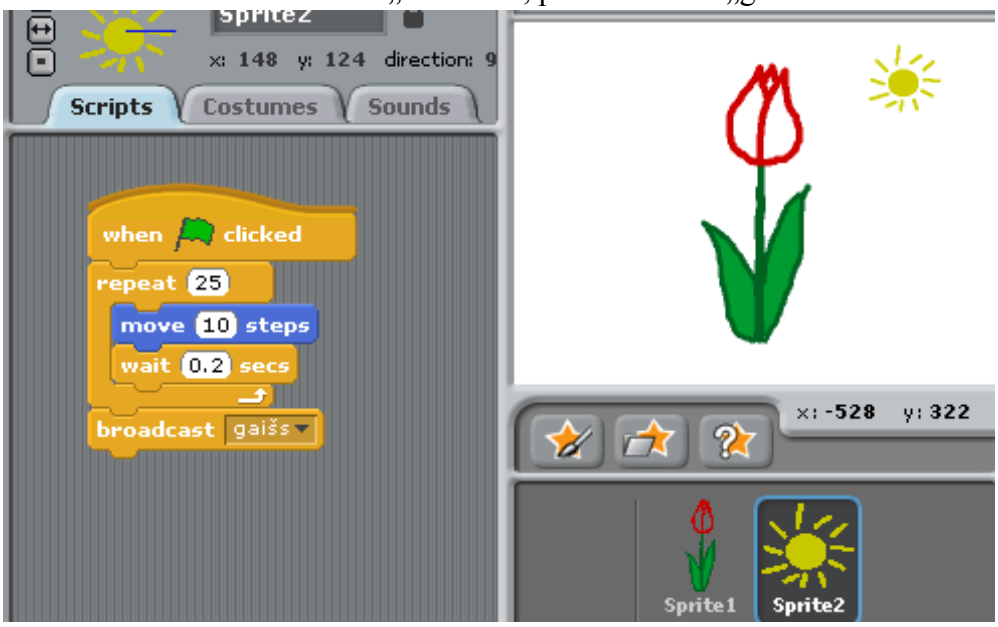
Prakstiskais darbs:
izveidot trīs objektus, izmantojot zīmēšanas rīkus un eksportēt uz bibliotēku.

10. nodarbība. Animācijas veidošana

Nodarbības mērķis: izveidot jaunu spraitus un tos animēt.

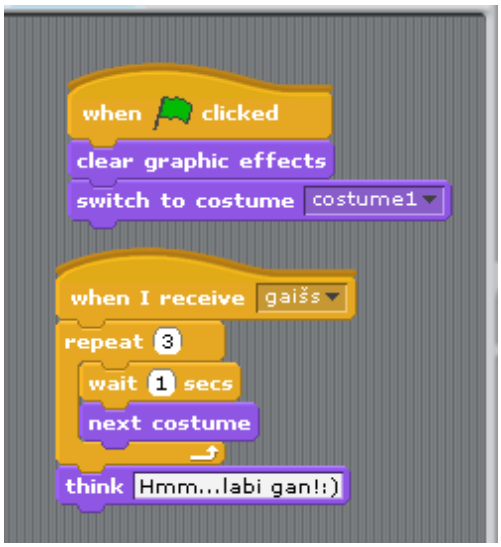
1) uzzīmēt ziedu;
2) sakopēt 4 objektus un ziedu papildināt.
3) uzzīmēt sauli;
4) izveidot programmu saulei.
5) ziedam izveidot programmas tekstu.

Programmas teksts saulei. Kad saule ir „uzlekusi", parādās teksts „gaišs".



Programmas teksts ziedam:

- notīrīt iepriekšējos grafiskos elementus un uzsākt ar 1.objektu;
- zieds sāk vērties vaļā, kad ir gaišs;
- komanda *next costume* ciklā nomaina spraitus un izveido animāciju.



Pārbaudīt izveidoto programmas teksu, noskatīties animāciju.

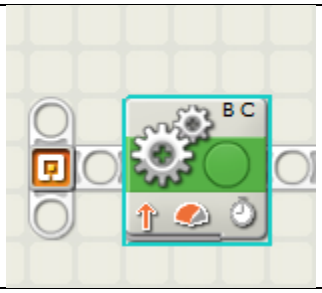
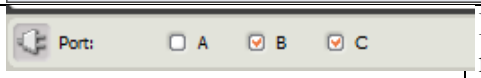
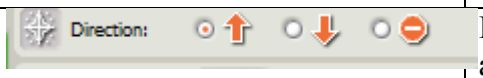
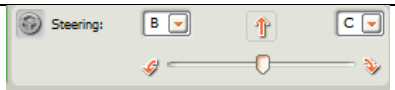
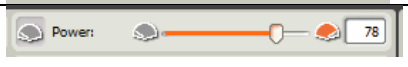
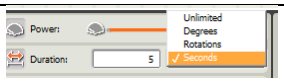
Patstāvīgais darbs:

1. izdomā sižetu;
2. uzzīmē vismaz divus objektus;
3. izveido animāciju!

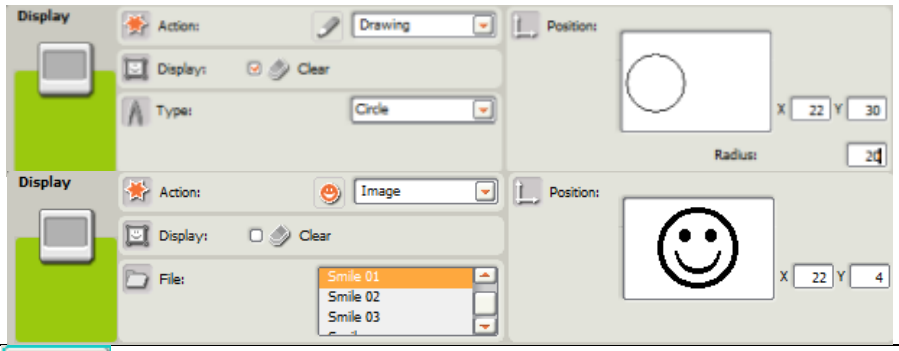
Lego Mindstorm NXT Mindstorm NXT2.0

Konstruktora *Lego Mindstorm NXT* Mindstorms NXT izmantošanai mācību procesā nepieciešamā materiāli tehniskā bāze ir: *Lego Mindstorm NXT* konstruktors, personālie datori ar grafikās programmēšanas vidi Mindstorms NXT.

Lego Mindstorm NXT Mindstorms NXT populārāko komandu apraksts:

<table>
<tr><td>Bloks</td><td colspan="2">Bloka darbība</td></tr>
<tr><td>Motora bloks</td><td> 
</td><td>Šis bloks kontrolē robota motoru un sinhronizē tā kustību. To izmanto, lai liktu robotam kustēties uz priekšu, atpakaļ pa taisnu vai liektu trajektoriju.</td></tr>
<tr><td>Motora konfigurācijas panelis</td><td></td><td>Konfigurācijas parametri:
• Ports;
• Virziens;
• Stūrēšana
• Jauda;
• Ilgums;
• Nākamā darbība.</td></tr>
<tr><td></td><td colspan="2"> 
</td></tr>
<tr><td>Porti</td><td> 
</td><td>Pēc noklusējuma B un C motori darbina riteņus;
A motoru izmanto papildierīču (ķepas, svira, u.c.) darbināšanai atkarībā no komplektācijas.
Atzīmētie motori tiek darbināti konkrētajā blokā.</td></tr>
<tr><td>Virziens</td><td> 
</td><td>Iespējamie kustības virzieni: uz priekšu, atpakaļ vai apturēt.</td></tr>
<tr><td>Stūrēšana</td><td> 
</td><td>Norāda kustības virzienu – taisni, pa kreisi, pa labi.</td></tr>
<tr><td>Jauda</td><td> 
</td><td>Jauda norāda cik procenti no motora maksimālās jaudas tiek izmantoti.
Ja jauda par mazu, nevar veikt kustību.</td></tr>
<tr><td>Ilgums</td><td> 
</td><td>Darbības ilgumam ir 4 parametri:
• Neierobežots;
• Grādi;
• Rotācija;
• Sekundes.</td></tr>
</table>

Gaidīšanas bloks 1		Gaidīšanas bloku Taimeris (*Wait /Time*) izmanto, lai izveidotu noteikta ilguma pauzi. To izdara norādot tekstlodziņā *Seconds* pauzes garumu veselās sekundēs
Gaidīšanas bloks 2		Gaidīšanas bloku Kontakts (*Wait /Touch*) izmanto, lai gaidītu kādu no trim Kontakta notikumiem (*Action*): sensora poga piespiesta (*Pressed*), sensora poga atlaista (*Released*), klikšķis uz sensora pogas (*Bumped*). Papildus ir jānorāda pie kura porta (Port: 1..4) ir pieslēgts sensors.
Gaidīšanas bloks 3		Gaidīšanas bloku Gaisma (*Wait /Light*) izmanto, lai gaidītu kamēr gaismas intensitāte ir lielāka (*Light > xx*) par izvēlēto līmeni vai ir mazāka (*Light < xx*). To var pārslēgt vai nu ar radiopogām slīdņa galos, vai nu ar izvelkamās izvēlnes palīdzību. Ja izvēles rūtiņa ir izvēlēta, tad tiek ģenerēts papildus apgaismojums un mērīta tiek atstarotās gaismas intensitāte. Intensitātes līmeni regulē ar slīdni diapazonā no 0 – 100%. Papildus ir jānorāda pie kura porta (Port: 1..4) ir pieslēgts sensors.
Gaidīšanas bloks 4		Gaidīšanas bloku Skaņa (*Wait /Sound*) izmanto, lai gaidītu kamēr skaņas signāla intensitāte ir lielāka (*Sound > xx*) par izvēlēto līmeni vai ir mazāka (*Sound < xx*). To var pārslēgt vai nu ar radiopogām slīdņa galos, vai nu ar izvelkamās izvēlnes palīdzību. Papildus ir jānorāda pie kura porta (Port: 1..4) ir pieslēgts sensors.
Gaidīšanas bloks 5		Gaidīšanas bloku Distance (*Wait /Distance*) izmanto, lai gaidītu kamēr attālums līdz šķērslim kļūst lielāks (*Distance > xx*) par izvēlēto vai arī mazāks (*Distance < xx*) par to. Lielāks/mazāks var pārslēgt vai nu ar radiopogām slīdņa galos, vai nu ar izvelkamās izvēlnes palīdzību. Attālumu līdz šķērslim var regulēt ar slīdņa palīdzību, vai arī ierakstīt tekstlodziņā Distance tiešā veidā. Attālumu var norādīt centimetros vai collās ar atbilstošās izvelkamās izvēlnes palīdzību. Attālums var būt robežās no 0 – 250 cm (0 – 100 collas). Sensors nodrošina precizitāti +/- 3cm. Papildus ir jānorāda pie kura porta (Port: 1..4) ir pieslēgts sensors.

Gaidīšanas bloks 6		Gaidīšanas bloku Krāsa (*Wait /Color Sensor Action=Color Sensor*) izmanto, lai gaidītu kamēr krāsa nav ārpus norādītā spektra (*Outside Range*) vai nav norādītajā spektra daļā (*Inside Range*). Ar slīdņa palīdzību izvēlas krāsas. Gaidīšanas bloku Krāsa (*Wait /Color Sensor Action=Light Sensor*) var izmantot arī kā gaismas sensoru, lai gaidītu līdz gaismas intensitāte pārsniedz norādīto (>) vai līdz tā samazinās zem norādītā līmeņa (*Inside Range*). To var pārslēgt vai nu ar radiopogām slīdņa galos. Ar slīdņa palīdzību izvēlas vajadzīgo gaismas intensitāti procentos. Ja izvēles rūtiņa ir izvēlēta, tad tiek ģenerēts papildus apgaismojums – sarkana, zaļa vai zila gaisma. Papildus ir jānorāda pie kura porta (Port: 1..4) ir pieslēgts sensors
Monitora bloks 1		Monitora bloks ļauj uz NXT (robota vadības bloks) monitora attēlot tekstu, attēlus un zīmējamus objektus. Teksta izvads (*Action = Text*). Monitorā izvada tekstlodziņā *Text* ievadīto tekstu – izvada līdz 16 simboliem. Ar parametru *Line* izvēlas rindu (1-8) uz monitora, kurā attēlos informāciju. Ja izvēles rūtiņa *Clear* ir izvēlēta, tad attēlošanas beigās teksts no ekrāna tiek nodzēsts.
Monitora bloks 2	Attēla izvads (*Action = Image*). Uz ekrāna izvada attēlu, kurš izvēlēts izvēlnē *File*. Lai esošajiem pievienotu citus attēlus, ir jāizmanto integrētais attēlu redaktors *Tools /Image Editor*. Ja izvēles rūtiņa *Clear* ir izvēlēta, tad attēlošanas beigās attēls no ekrāna tiek nodzēsts. Attēla izvads (*Action = Drawing*). Uz ekrāna izvada zīmēto objektu (punktu, līniju, riņķa līniju), kuru norāda ar izvelkamās izvēlnes *Type* palīdzību. Ja izvēles rūtiņa *Clear* ir izvēlēta, tad attēlošanas beigās attēls no ekrāna tiek nodzēsts. 	
Skaņas bloks 1		

	 Skaņas faila atskaņošana (*Action = Sound File*). Tiek atskaņots sarakstlodziņā *File* izvēlētais skaņas fails. Skaņas stiprumu var regulēt ar slīdņa *Volume* palīdzību. Ja izvēles rūtiņa *Repeat* ir izvēlēta, tad atskaņošana tiek visu laiku atkārtota. Ar kontroles radiopogu *Play* startē atskaņošanu, ar radiopogu *Stop* – aptur atskaņošanu. Ja ir izvēlēta izvēles rūtiņa *Wait*, tad tiek sagaidītas atskaņošanas beigas pirms sāk pildīt nākamo bloku. Jaunus skaņas failus esošajam sarakstam var pievienot, izmantojot izvēlni *Tools /Sound Editor*. Var pievienot gan veselu failu, gan no vesela izgriezt tikai kādu fragmentu un to pievienot skaņas failu sarakstam.
Skaņas bloks 2	  Skaņas toņa atskaņošana (*Action = Tone*). Tiek atskaņota tekstlodziņā *Note* izvēlētā nots tik sekundes cik norādīts tekstlodziņā *for*. Skaņas stiprumu var regulēt ar slīdņa *Volume* palīdzību. Ja izvēles rūtiņa *Repeat* ir izvēlēta, tad atskaņošana tiek visu laiku atkārtota. Ar kontroles radiopogu *Play* startē atskaņošanu, ar radiopogu *Stop* – aptur atskaņošanu. Ja ir izvēlēta izvēles rūtiņa *Wait*, tad tiek sagaidītas atskaņošanas beigas pirms sāk pildīt nākamo bloku.

1.nodarbība. Iepazīšanās ar *Lego Mindstorm NXT* NXT 2.0 vidi

Nodarbības mērķis: iepazīšanās ar robotu, robotu nozīmi un pielietojumu ikdienā, *Lego Mindstorm NXT* Mindstorm NXT2.0 programmatūru.

Prezentācijas noskatīšanās. Diskusija.

Iepazīšanās ar NXT vidi.

Pirms uzsākt darbu ar *Lego Mindstorm NXT* Mindstorm NXT2.0 jāuzbūvē robots. Tas prasa zināmu laiku, attiecīgi no sarežģītības pakāpes, bet tā ir atsevišķa nodarbība. Pirms sākt veidot programmu ir jāpārliecinās kādi sensori ir robotam, tas nozīmē – kādus sensorus var iekļaut programmas tekstā.

Uzdevums. Izveidot nelielu algoritmu :

1. kustība taisni 4 rotācijas;
2. kustība pagriezienā ka kreisi 6 rotācijas;
3. kustība taisni 4 rotācijas;
4. kustība pagriezienā ka labi 6 rotācijas;
5. kustība taisni 4 rotācijas;
6. 6 rotācijas ar izslēgtu B motoru
7. kustība taisni 4 rotācijas;
8. 6 rotācijas ar izslēgtu C motoru

Piebilde: B un C portiem ir pieslēgti motori, kas atbild par rotāciju.

Izveidoto programmu saglabāt un notestēt ar robotu. Kļūdas gadījumā labot kļūdu un testēt vēlreiz.

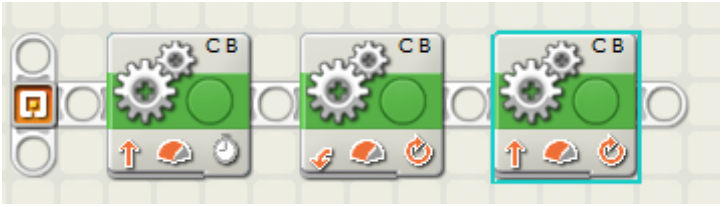
2.-3.nodarbība. Iepazīšanas ar lineāra algoritma veidošanu *LEGO MINDSTORM NXT* vidē

Nodarbības mērķis: lineāru algoritmu veidošana NXT vidē.

Nepieciešams sensors *touch*.

1) Sastādīt algoritmu: robots iet uz priekšu 5 sekundes, pagrieziens pa labi, kustība uz 10 rotācijas (motora apgriezieni).
2) Sastādīt algoritmu: robots iet pa iedomātu kvadrātu;
3) Sastādīt algoritmu: robots iet, līdz ceļā parādās šķērslis, atkāpjas 1 sekundi, pagriežas.

1.programmas teksts – blokā ir norādes vai kustība ir uz priekšu, vai atpakaļ, vai pagrieziens, kurā virzienā, vai laiks, vai rotācijas.



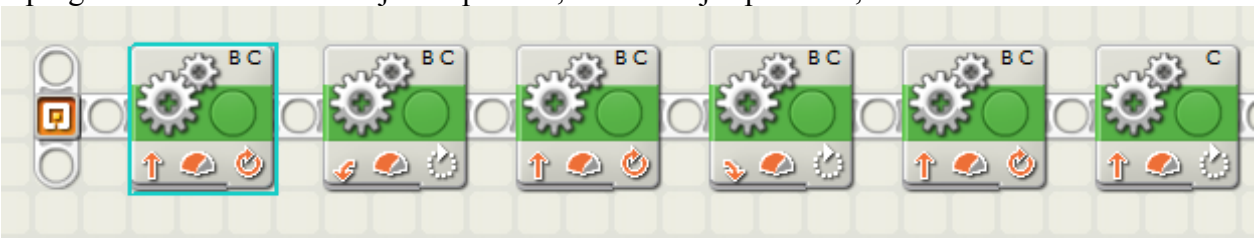
1.boka iestatījumi

2.bloka iestatījumi

3.bloka iestatījumi

Kustība notiek secīgi viena darbība seko otrai, tas nozīme, tas ir lineārs algoritms.

2.programmas teksts: 4 rotācijas uz priekšu, 300 rotācijas pa kreisi, atkārtot 4 reizes.



3. programmas teksts

1.bloks – kustība neierobežota;

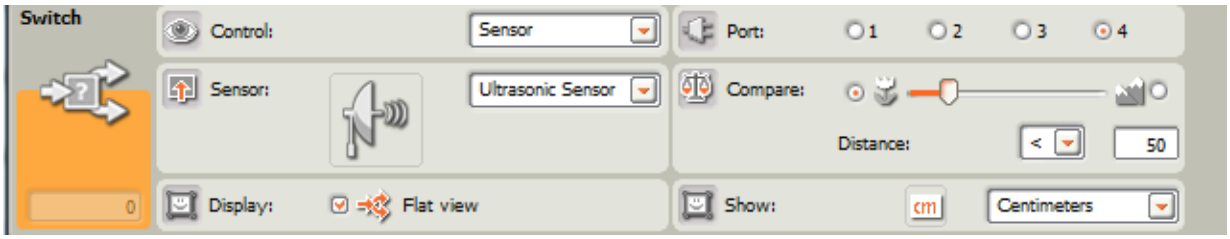
2.bloks – sensors *touch*.

3.bloks - atkāpjas un 4. bloks - pagriežas.
Analizējam algoritmu: kustība, saskare ar šķērsli, atkāpties, pagriezties. Kāds tas ir algoritms? Lineārs.
Uzzīmēt visiem uzdevumeim blokshēmu!
Praktiskais darbs.
1)sastādīt programmu lineāram algoritmam;
2)ielādēt programmu robotā, testēt;
3)izlabot kļūdas;
4)vēlreiz testēt, līdz kļūdas novērstas;
5)veidot aprakstu, algoritmu uzzīmēt blokshēmas veidā;
6)demonstrēt projektu, pamatojot ar stāstījumu.

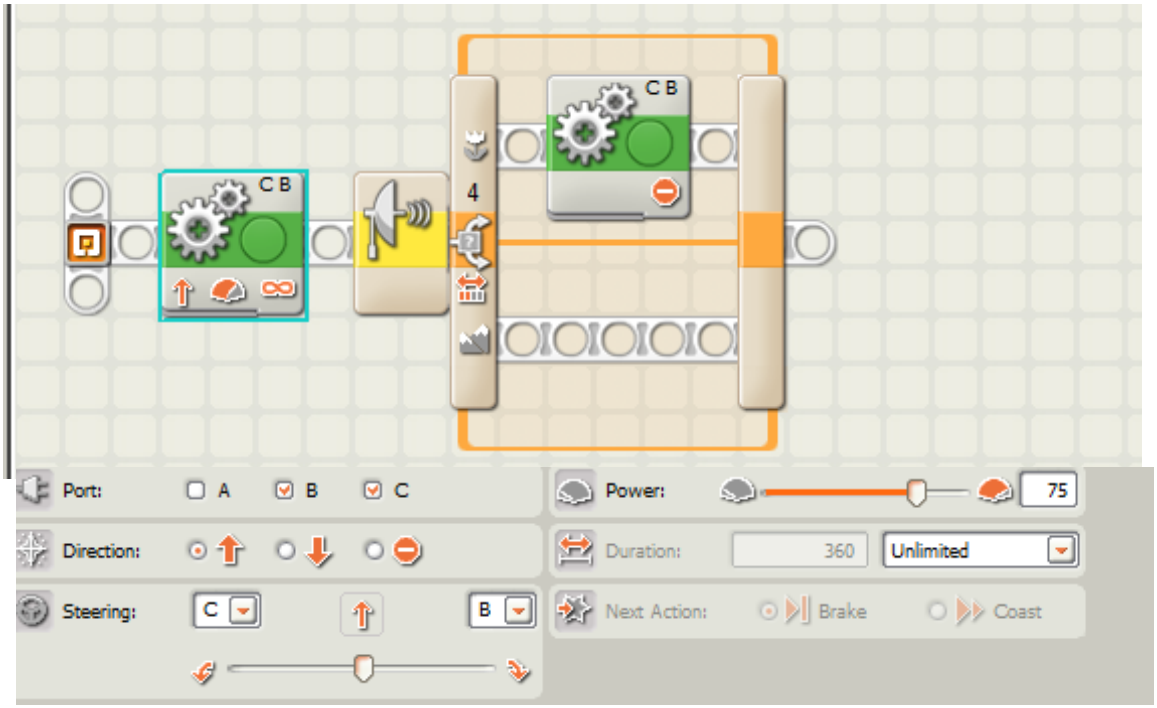
4.-5.nodarbība. Sazaroti algoritmi

Sastādīt algoritmu: *Lego Mindstorm NXT* robots iet, ja ceļā parādās šķērslis, tad apstājās 10 soļus līdz šķērslim.

Lai robots uztvertu šķērsli, nepieciešams sensors *Ultrasonic Sensor*.



Ielikt bloku – sazarojums, izvelēties pareizo sensoru *Ultrasonic Sensor*. Nosacījums izpildās .

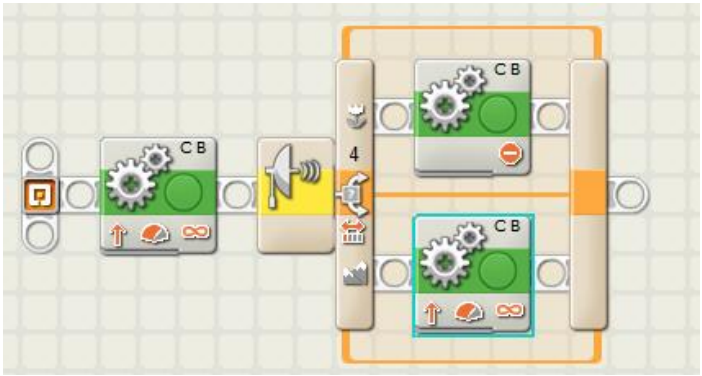


Papildināt programmu ar kustības blokiem

Pirmajam kustības blokam uzlikt neierobežotu kustību. Otram blokam – pie „Jā" zara – ir šķērslis, tātad jāapstājas. Trešajam blokam likt bezgalīgu kustību.

Analizēt algoritmu: kustība, ja attālums līdz šķērslim ir mazāks par 50 cm robots apstājas, ja nē – turpina kustību. Kāds tas ir algoritms? Sazarots.

Uzzīmēt uzdevumam blokshēmu!



Praktiskais darbs.

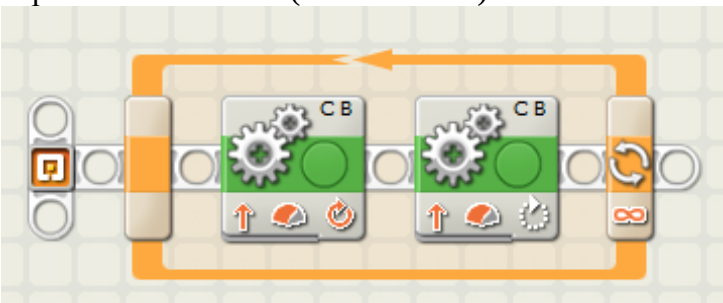
1)sastādīt programmu sazarotam algoritmam;

2)ielādēt programmu robotā, testēt;
3)izlabot kļūdas;
4)vēlreiz testēt, līdz kļūdas novērstas;
5)veidot aprakstu, algoritmu uzzīmēt blokshēmas veidā;
6)demonstrēt projektu, pamatojot ar stāstījumu.

6.-7.nodabība Cikla realizācija NXT vidē

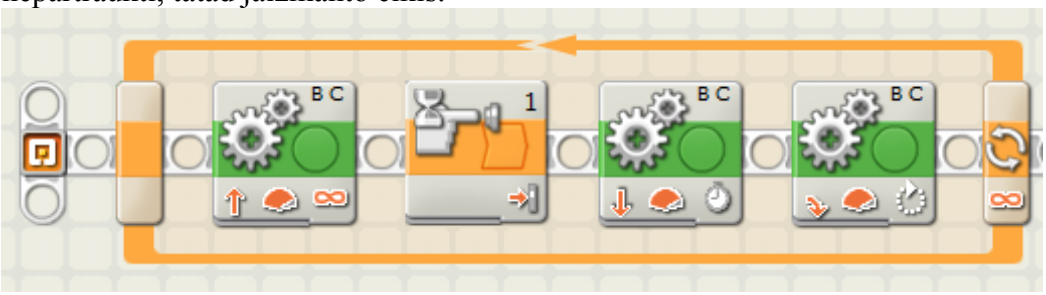
Cikls ir jēdziens, ar ko apzīmē vairākkārtīgu kādas instrukciju kopas izpildīšanu. Lai robots pārvietotos pa iedomātu kvadrātu var veidot lineāru algoritmu, bet tas ir neracionāli. Robots iet tikai tik tālu, cik tas ir ieprogrammēts. Ar cikla palīdzību par nodrošināt nepārtrauktu darbību atkārtošanu, respektīvi – izpildi.

Atvērt 2.uzdevumu – robots iet pa iedomātu trajektoriju, ievietot ciklu, atstāt tikai nepieciešamos blokus (1. un 2. bloku).



1.bloks nodrošina kustību uz priekšu, 2.bloks – pagrieziens pa noteiktu leņķi.

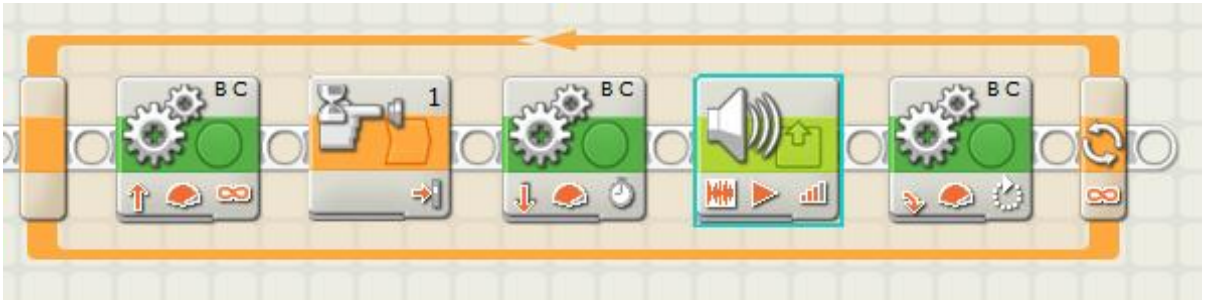
Uzdevums. Sastādīt programmu: robots pārvietojas uz priekšu (kustība neierobežota), līdz šķērslim, atkāpjas 1 sekundi, pagriežas pa brīvi izvēlētiem grādiem. Darbība notiek nepārtraukti, tātad jāizmanto cikls.



Pārliecināties, vai izveidotā programma strādā (notestēt uz robota).

Uzzīmēt algoritma blokshēmu!

Papildināt programmu, piemēram, pievienot skaņu – robots saduroties ar šķērsli atvainojas.



Patstāvīgais darbs.

1)sastādīt programmu cikliskam algoritmam;
2)ielādēt programmu robotā, testēt;
3)izlabot kļūdas;
4)vēlreiz testēt, līdz kļūdas novērstas;
5)veidot aprakstu, algoritmu uzzīmēt blokshēmas veidā;
6)demonstrēt projektu, pamatojot ar stāstījumu.

8.-10.nodabība Projektu veidošana, patstāvīgs darbs NXT vidē.

Skolēni meklē jau gatavus projektus. Būvē robotus (grupu darbs, atkarīgs no skolēnu un robotu skaita)