Невенка Спалевић

ПРОГРАМСКИ ЈЕЗИК СКРЕЧ (Scratch), 2. део

Како се користи и чему служи – представљање кроз примере пројеката

У претходном броју говорили смо о интерфејсу програма Скреч, ликовима, сцени и скриптама – сценаријима који описују њихово понашање. У информатици се уместо термина сценарио и скрипта користи термин – алгоритам. У овом броју изложићемо како се у оквиру предмета Информатика и рачунарство уз коришћење програмског језика Скреч може увести појам алгоритма.

Шта је алгоритам

Термин алгоритам настао је као латински превод имена персијског математичара и астронома Мухамеда ибн Мусе ал Хорезмија (арап. Myhammad ibn-Musa al-Khwàrizmi), који је живео у IX веку на подручју данашњег Узбекистана. У преводу с арапског језика то би значило: Мухамед син Мусе из Хорезма. Ал Хорезми је разрадио и у посебној књизи описао правила за извршавање аритметичких операција са декадним бројевима. Арапски оригинал те књиге је изгубљен, али постоји њен превод на латински из XII века. У том преводу испред сваког правила пише „*Dixit Algorizmi*“– „Алгоризми је говорио“, односно крај имена ал Хорезми претворен је у Алгоризми. У другим латинским текстовима име је претворено у Algorithmus – Алгоритам. Временом је заборављено да је Алгоритам име човека, па се наслов над правилима „*Алгоритам је говорио*“претворио у „алгоритам гласи“. На тај начин су се правила почела називати алгоритмима.

Појам алгоритма је познат вековима, али посебан значај добија тек појавом рачунара. У прво време су се под термином алгоритам подразумевала правила за вршење четири основне аритметичке операције, а данас овом термину одговара далеко шире значење. Прелазак улице, припрема хране по рецепту, обављање телефонског разговора, решавање математичког задатка – све то се своди на извршавање појединих елементарних операција задатим редоследом. Елементарне операције се у алгоритмима називају **команде**, **кораци** или **инструкције** (лат. instructio – упутство). Често се уместо речи редослед (корака) користи термин низ, што сугерише да се инструкције извршавају једна за другом, секвенцијално, а то није увек коректно. Наиме, рачунари могу паралелно да решавају више задатака и истовремено извршавају више инструкција, слично као што се више возила

истовремено креће паралелним коловозним тракама. И људи могу да извршавају више задатака истовремено, на пример да седе, читају и слушају музику. У Скречу ликови такође могу паралелно да извршавају више скрипти, а уз то више ликова на сцени у исто време може да ради сваки по свом сценарију понашања.

Можемо рећи да сваки скуп разумљивих и прецизних упутстава како да се реши постављени задатак представља алгоритам, али ако желимо да будемо до краја коректни, морамо додати „у интуитивном смислу“. Наиме, у математици се и надаље алгоритми изучавају на систематичан теоријски начин. Математичка теорија алгоритама бави се могућим облицима алгоритама и истражује својства самих алгоритама. С друге стране, са становишта употребљивости, алгоритам није објекат изучавања већ се користи за јасно *утврђивање правила достизања неког постављеног циља*. Међутим, потребно је ипак утврдити нека основна својства алгоритама и установити када се неки поступак може назвати алгоритмом.

Својства алгоритама

Шта мислите да ли правило „Кад излазиш угаси светло“ представља алгоритам? Да, мада врло примитиван. Али правило „За време кретања тротоаром држите се леве стране“ то није, јер има непрекидан карактер. Алгоритмима можемо описати једино процесе који имају дискретан карактер, који се одвијају кроз низ засебних дејстава. Ова дејства описују се алгоритамским корацима који се извршавају један за другим. Сваки алгоритамски корак састоји се из два упутства: шта треба урадити и који алгоритамски корак треба извршти за њим. Дакле, алгоритамски корак не само што прецизира операцију коју треба извршити, него једнозначно указује и на следећи алгоритамски корак. Ово својство алгоритама зове се **дискретност**.

Алгоритам треба да буде разумљив за корисника. Другим речима, свако упутство које се задаје алгоритамским корацима треба да буде такво да извршилац алгоритма уме да га уради. Ако, рецимо, тражите од ученика трећег разреда основне школе да квадрира неки број, он ту операцију вероватно неће моћи да уради, али ако захтевате да помножи број самим собом, успеће да реши задатак. Ово својство назива се **елементарност алгоритамског корака**.

Алгоритам увек мора да се заврши после коначног броја корака. Ово особину зовемо **коначност**. При том, алгоритам увек мора да доведе до решења задатка. Приметимо да и утврђивање чињенице да задатак нема решења представља у ствари решење проблема. Ово својство алгоритма зове се **резултативност**. Алгоритми имају још једну важну карактериситику – **масовност**. Та особина подразумева да се помоћу једног истог алгоритма може решавати читава класа задатака са различитим улазним величинама. Својство масовности значајно увећава практичну вредност алгоритама. Уз дискретност, резултативност и масовност, оно што суштински одређује алгоритме је својство **детерминисаности**. Наиме, за исте улазне величине алгоритам увек мора дати исте резултате.

Скуп команди извршиоца

Као што смо рекли, кораци алгоритама не извршавају се сами по себи, њих обавља **извршилац алгоритма**, који разуме алгоритам и зна тачно да обави сваки корак алгоритма. Сви објекти – и жива бића и технички уређаји могу бити извршиоци алгоритама. Зашто машина за прање веша не може да пече колаче, а фрижидер да пере веш? Зато што је сваки објекат у стању да изврши само ограничени скуп команди. Машина за прање веша у стању је да прихвати воду за прање, загрева воду, центифугира – дакле да извршава команде прања, али не и да извршава команде печења.

Скуп команди које може да извршава објекат назива се скуп команди извршиоца. Рачунари су универзални извршиоци алгоритма захваљујући уграђеном скупу инструкција (инструкција машинског језика) на које се могу свести различити програми – упутства за решавање проблема из практично свих области људске делатности.

У Скречу постоје две врсте извршилаца алгоритама: сцена и ликови-спрајтови (слика 1).

Слика 1. Извршиоци алгоритама у Скречу

Ликови могу да извршавају више од 120 команди, а сцена нешто мање – 85. Овај скуп омогућава извршиоцима да реализују мноштво различитих алгоритама. Све расположиве команде чувају се у левом прозору програма и организоване су у 8 група. Блокови команди сваке групе имају карактеристичну боју.

Motion	кретање (тамно плава)	Control	управљање (жута)
Looks	изглед (љубичаста)	Sensing	сензори (светло плава)
Sound	звук (пурпурно црвена)	Numbers	оператори (светло зелена)
Pen	оловка (тамно зелена)	Variables	променљиве (наранџаста)

Запис и структура алгоритама

Алгоритам можемо мање или више прецизно задати речима. Међутим, много бољи увид у структуру и редослед извршавања инструкција даје графички запис

алгоритма. Графичка шема која својим симболима указује на природу појединих алгоритамских корака назива се **блок дијаграм**. У блок дијаграмима се користе специјални симболи за запис:

- почетног и завршног корака,
- улазно/излазних корака,
- корака обраде и
- корака доношења одлуке.

Основни циљ блок дијаграма је да укаже на редослед извршавања појединих алгоритамских корака. Линија која у блок дијаграму указује на следећи алгоритамски корак зове се *повезница*. Графички запис алгоритма, поред тога што омогућава краћи и јаснији запис алгоритма него писаним језиком, даје прегледну везу између детаља и целине алгоритма и омогућава лако откривање грешака у његовој структури.

Скрипте у Скречу изгледају као запис алгоритама блок дијаграмима.

Инструкције у програмском језику Скреч задају се превлачењем одговарајућих блокова у област скрипте. Као што повезнице спајају алгоритамске кораке и указују на редослед њиховог извршавања, тако се блокови инструкција међусобно повезују тако што се слажу један преко другог, при чему се испупчење једног блока постави у удубљење другог (слика 2а). Овакве блокове који одговарају корацима обраде звaћемо надаље **стек блокови**, а њихова карактеристика је да имају удубљење на врху и испупчење на дну. Блокове који одговарају почетном алгоритамском кораку чија је карактеристика да на њих не указује ни један алгоритамски корак у Скречу називамо „**капе**“. Овај тип блокова има заобљен врх (слика 2б) и поставља се на врх стека, а повезује се са наредним блоковима преко испупчења на дну. „Капе“ дефинишу који догађај треба да наступи да би се извршио стек блокова испод њих. Тај догађај може да буде, на пример, притисак на неку типку тастатуре, добијање одговарајуће поруке или клик на лик.

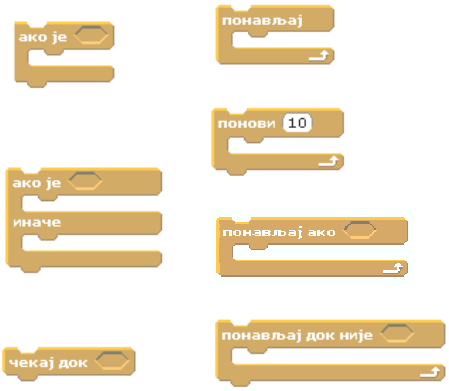
Слика 2. Примери блокова инструкција у Скречу

Блокови који одговарају завршном алгоритамском кораку имају удубљење на врху, али немају испупчење на дну. Како у Скречу једном објекту може бити придружено више скрипти које се могу паралелно извршавати, постоји и блок који завршава извршавање свих покренутих скрипти (слика 2ц). Улазни подаци примају се у Скречу помоћу групе блокова који се зову **сензори** (слика 2д). Податак

који корисник унесе чува се у предефинисаној променљивој **одговор**. Вредност променљиве одговор може се доделити ма којој променљивој или листи коју је корисник креирао. Скреч сам утврђује тип променљиве (реални, целобројни или стринг) на основу операције која се над њом извршава. Како Скреч омогућава програмирање управљано догађајима, многи од блокова сензора умећу се у кораке гранања. Ако је наступио догађај који се прати сензором, извршава се једна група корака – тзв. „да-грана“, а ако није, онда се кораци „да-гране“ не извршавају. Блокови који одговарају логичким променљивим су облика шестоугла и могу се уметнути само у блокове који имају отвор одговарајућег облика. Блокови који одговарају променљивим у Скречу се називају **репортери**.

Резултат алгоритма може бити нека акција коју реализује извршилац алгоритма, на пример кретање, свирање, промена изгледа, али и издавање података. Многи од блокова инструкција који омогућавају приказ резултата алгоритма налазе се у групи **изглед** (слика 2д).

Коначно, блокови који одговарају корацима гранања, налазе се у групи инструкција **управљање** (слика 3).



Слика 3. Блокови који омогућавају гранања и циклусе

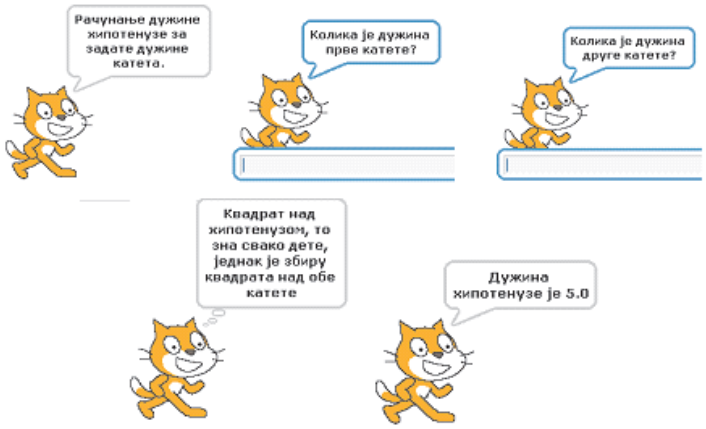
Из блок дијаграма се јасно може видети структура алгоритма, односно логика програма. Програми разгранате структуре у Скречу се реализују помоћу блокова непотпуног и потпуног гранања и условне паузе (прва колона слике 3). Условни оператори могу да се умећу један у други и тако реализују гранања произвољне сложености.

У Скречу се могу реализовати 4 врсте циклуса: безусловни, бројачки, с предусловом и с постусловом (друга колона слике 3).

Примери програма у Скречу

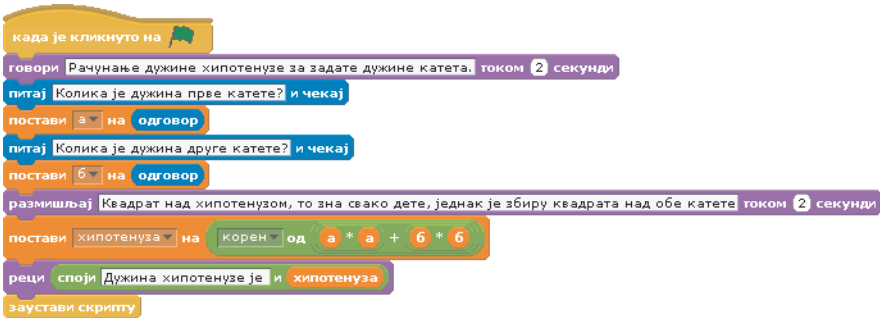
Задатак 1. Написати програм којим се израчунава хипотенуза правоуглог троугла на основу задатих катета.

Прво решење. У овом решењу тражење улазних података и саопштавање резултата поверићемо неком лику – у нашем примеру мачку, заштитном лику



Слика 4. Прихватање улазних података и издавање резултата које води лик

Скреча. Читав поступак изгледаће као стрип (слика 4) – прво мачак саопштава шта ради програм, затим тражи улазне податке, онда „размишља“ како ће израчунати резултат и на крају га саопштава. Како можете видети из скрипте придружене мачку, програм има просту линијску структуру (слика 5).

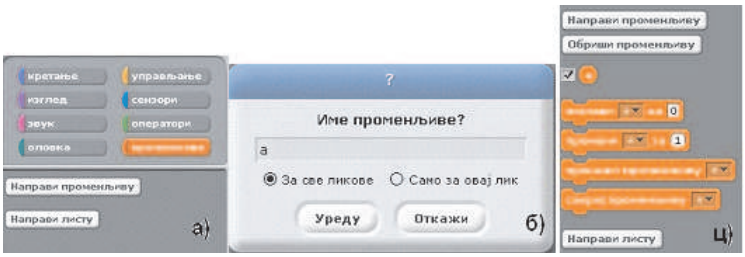


Слика 5. Скрипта за израчунавање дужине хипотенузе придружена лику

Задржимо се мало на начину креирања променљивих у програму.

Креирање и коришћење променљивих

Избором групе **променљиве** у палети блокова појављују се дугмићи „*Направи променљиву*“ и „*Направи листу*“ (слика 6а). Кликом на дугме „*Направи променљиву*“ отвара се дијалог за креирање нове променљиве (слика 6б). Променљива може бити локална, само за скрипте придружене објекту у фокусу (лику или сцени) или глобална коју могу да користе сви објекти пројекта. Уписом имена променљиве и кликом на дугме „*У реду*“ у палети блокова појављују се 4 инструкције и репортер са именом променљиве уз који стоји дугме за потврду (слика 6ц).



Слика 6. Креирање променљиве

Репортери се појављују у два облика и могу да се поставе само у отворе истог облика. Репортери са заобљеним крајевима (елипсе) садрже бројеве и могу да се поставе у блокове који имају заобљену белину. Репортери шестоугаоног облика садрже логичке податке (тачно или нетачно) и могу да се поставе у блокове са одговарајућим отвором. Неки репортерски блокови имају и поље за потврду. Ако се кликне на поље за потврду, на сцени се појављује монитор у коме се приказује текућа вредност репортера. Промене вредности репортера аутоматски се приказују на монитору. Ово је нарочито корисно приликом креирања пројекта, специјално ако се користи режим рада корак по корак. Монитор може да приказује вредности репортера у више различитих формата:

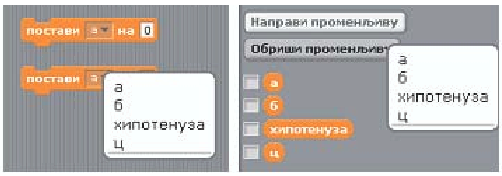
а) мали приказ вредности са именом репортера,

б) велики приказ вредности без припадајућег имена,

ц) са клизачем који омогућава промену вредности репортера (слика 7).

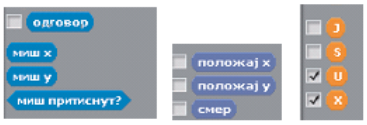
Слика 7. Различити прикази репортера

Прелазак из једног у други формат постиже се двокликом на монитор. Формат са клизачем може се користити једино за променљиве које креира корисник. Десним кликом на клизач подешавају се минимална и максимална вредност. У програму се користе три нумеричке променљиве: **а, б** и **хипотенуза**. Када се у програму креира више променљивих у наредби за доделу вредности се избором из падајућег менија бира којој од њих желимо да доделимо вредност. Кликом на дугме „*Обриши променљиву*“ добијамо у падајућем менију списак свих коришћених променљивих из којег бирамо ону коју желимо да обришемо (слика 8).



Слика 8. Додела вредности и брисање променљиве

Променљивим **а** и **б** вредност смо додели помоћу уграђене променљиве **одговор** која прима улазни податак који уноси корисник после инструкције за постављање питања. Скреч користи више предефинисаних променљивих које указују на текућу позицију миша, позицију центра лика и слично (слика 9).



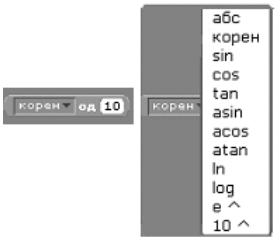
Слика 9. Предефинисане променљиве из група сензори, кретање и корисничке променљиве

Вредност променљивој **хипотенуза** доделили смо као вредност израза **корен(а*а+б*б)**. Бројевни израз формира се преношењем одговарајућег блока из групе **оператори**, у нашем случају блока за сабирање у чије се отворе уписују одговарајући подаци, променљиве или изрази. Променљиве се уписују тако што се одговарајући репортер преноси из палете променљиве и поставља у отвор, а изрази тако што се одговарајући израз преноси из палете оператори. У нашем случају се у празна поља блока за сабирање постављају блокови за множење, од којих се у оба поља првог блока умећу репортери променљиве **а**, а у оба поља другог блока репортери променљиве **б**. Затим се формирани израз **а*а+б*б** поставља у празно поље блока за израчунавање функције **корен** који се такође узима из групе оператори (слика 10). На крају се цео израз убацује у празнину инструкције постави (вредност променљиве хипотенуза) која се узима из групе блокова променљива.

Слика 10. Формирање вредности израза коришћењем блокова из групе оператори

Слично израчунавању функције корен могли смо изабрати неку другу нумеричку функцију избором из падајућег менија блока функција из групе оператори (слика 11).

Саопштавање резултата остварили смо повезивањем стринга „Дужина хипотенузе је“ и вредности променљиве хипотенуза. Напомињемо да се мора користити репортер променљиве из палете блокова променљиве, а не текст са именом променљиве.



Сл. 11

Уграђене нумеричке функције могу се изабрати из падајућег менија блока са дна групе оператори

Типови података у Скречу

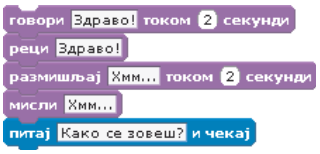
У Скречу се могу користити нумерички и логички подаци, као и стрингови и листе.

Нумерички подаци се састоје из низа цифара којима може претходити знак минус (−) за означавање негативног броја, и између којих се може навести тачка да би раздвојила целобројни од децималног дела броја. То су означени цели и реални бројеви. Нумерички подаци се уписују у овална поља блокова.

Коришћењем блокова репортера из групе **оператори**, над бројевима се могу извршавати основне аритметичке операције и математичке функције, може се генерисати случајни број из задатог интервала, заокруживати реалан број на ближи цео број, израчунати остатак при целобројном дељењу.

Логички подаци могу имати само две вредности: *тачно* и *нетачно*. Логички оператори и логички подаци користе се за програмирање циклуса и гранања унутар блокова са „устима“ у облику спљоштеног слова C у која можете да убаците друге стек блокове. Они омогућавају да се направи илузија разумног понашања ликова.

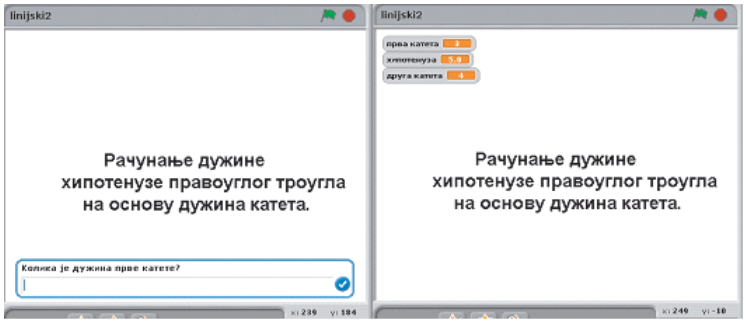
Стрингови су ниске произвољних симбола (слова, цифре, интерпункцијски знаци), ма које дужине, које се могу написати коришћењем тастатуре. Стрингови се уписују у правоугла поља блокова. То су најчешће различите поруке које се могу приказати помоћу четири блока из групе **изглед** и једног блока из категорије **сензори** (слика 12).



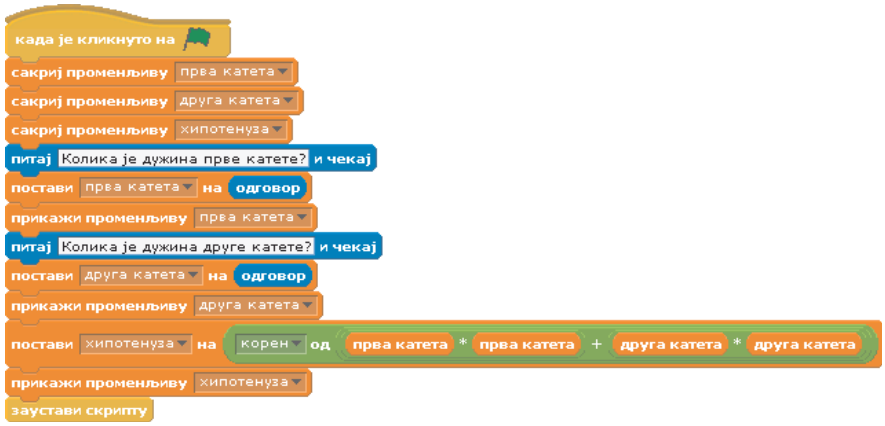
Слика 12. Стрингови у пољима блокова за саопштавање порука

Са стринговима се могу извршавати операције, као са бројевима, али таквих операција је значајно мање – свега 3: повезивање стрингова, утврђивање броја симбола у стрингу и приказивање симбола који се налази на задатој позицији у стрингу. *Листе у Скречу омогућавају рад са низовима података.*

Друго решење задатка 1. Друга стратегија за прихватање улазних података и приказивање излазних резултата је придруживање скрипте сцени. Задатак се поставља као позадина сцене, сцена „захтева“ улазне податке, резултат се очитава на монитору података (слика 13). Улазне податке можемо уносити коришћењем клизача на монитору променљивих, али то није увек прецизно. Приказивање монитора променљивих можемо програмским путем контролисати слично као што ликове можемо сакривати или приказивати. У скрипти која следи користимо променљиве **прва катета**, **друга катета** и **хипотенуза**, а пошто сцена по природи ствари не може да говори, резултат приказујемо преко монитора променљиве хипотенуза. Скрипта придружена сцени приказана је на слици 14.



Слика 13. Прихватање улазних података и издавање резултата које води сцена

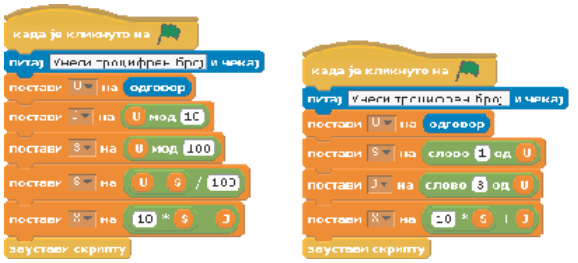


Слика 14. Скрипта за израчунавање дужине хипотенузе придружена сцени

Задатак 2. Написати програм којим се избацује цифра десетица у природном троцифреном броју x.

Прво решење – коришћењем нумеричких података. У програму користимо следеће променљиве: U – троцифрени улазни податак, J – цифра јединица, S – цифра стотина, X – двоцифрени резултат. Издвајање цифре јединица остварујемо коришћењем оператора mod ($J = U \mod 10$). Пошто не располажемо оператором за целобројно дељење, до цифре стотина долазимо тако што од улазног податка одузмемо ($U \mod 100$) и тај број поделимо са 100. На пример, за $U = 325$ вредност израза ($U \mod 100$) износи 25, а израза $U - (U \mod 100)$ износи 300, па дељењем са 100 добијамо 3, односно цифру стотина. Резултат се добија по формули $X = S * 10 + J$.

Друго решење – коришћењем стрингова. Уместо целобројног дељења и оператора mod до цифара улазног податка можемо доћи коришћењем оператора за издвајање знака на задатој позицији стринга. Захваљујући чињеници да Скреч сам реализује конверзије из нумеричких података у стрингове и обрнуто, друго решење је једноставније. Скрипте за оба решења задатка 2 приказане су на слици 15.



Слика 15. Скрипте за решења задатка 2.

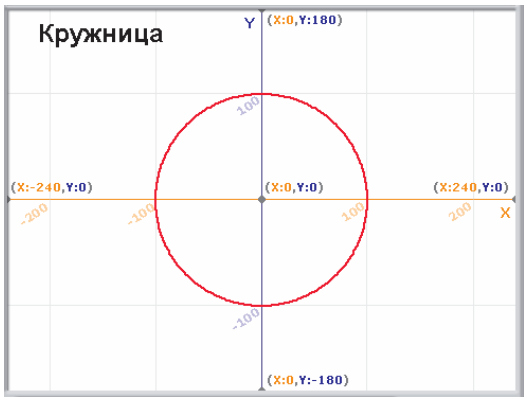
Задатак 3. Ако се молекул сумпорне киселине H_2SO_4 састоји из 2 атома водоника, једног атома сумпора и 4 атома кисеоника, написати програм који одређује максималан број молекула сумпорне киселине који се може формирати од датих H атома водоника, S атома сумпора и O атома кисеоника.

Решење је приказано на слици 16.

Слика 16. Скрипта са инструкцијама гранања

Задатак 4. Написати програм који црта кружницу са центром у средини сцене и полупречником 100 тачку по тачку.

Решење. На слици 17 можемо видети изглед сцене по завршетку скрипте придружене лику који у циклусу који се понавља 360 пута оловком спаја тачке на кружници. Почиње се од тачке с координатама $(100, 0)$ која са x осом заклапа



Слика 17. Кружница нацртана програмом чија је скрипта приказана на следећој слици

угао од 0 степени, па се иде на крај полупречника круга који са x осом заклапа угао од 1 степена и тако редом све до повратка у тачку (100,0).

Алгоритам цртања приказан је на слици 18.

Слика 18. Скрипта програма који црта кружницу са цикличном структуром

Напомена. У свим примерима приказаним у овом броју коришћена је српска локализација програмског окружења Скреч. Локализацију је урадила Јелена Хаџи-Пурић.

Математичка гимназија, Београд