Мр Јелена Хаџи-Пурић

АЛГОРИТМИЗАЦИЈА ТАКМИЧАРСКИХ ЗАДАТАКА И РЕШАВАЊЕ У ПРОГРАМСКОМ АЛАТУ BYOB И SNAP

1. Увод

У претходним бројевима Наставе математике било је речи о програмирању у Скречу и начину да се програмирање учини доступним и довољно занимљивим за децу.

Методологија развоја Скреча доста црпи из методологије конструктивистичког приступа у учењу програмирања и програмерских техника [1]. Seymour Papert, творац програмског језика Logo, можда је и најбоље објаснио конструктивистичку идеју у својој књизи *Mindstorms: Children, Computers, and Powerful Ideas*: Учење програмерских техника и програмерских идеја деци је могуће само ако им омогућимо боравак у "mathland" (земљи математике), на исти начин као што је природно учити француски језик живећи у Француској.

Као надоградња програмског језика Скреч, настало је и развојно окружење BYOB (offline и online верзија) и SNAP (online верзија). И BYOB и SNAP су пројектовани тако да се не захтева коришћење тастатуре, јер су основни конструкти језика доступни као визуелни блокови који се превлаче са менија. Основна идеја водиља код SNAP-а и BYOB-а је да се Скреч понуди озбиљнијим корисницима као што су ученици који се спремају за такмичења из програмирања и студенти. У том смислу, оба језика нуде и рад са напредним структурама података и алгоритмима како би ученици могли да се изразе креативније у свету нових технологија. Оба језика су заправо настала кад се уочило да постоји велика потреба у средњим школама и на факултетима за средством за „програмерско истраживање и самоизражавање“ које ће снажније привући ученике програмирању без обзира да ли су изабрали информатику као будуће занимање.

Сви корисници Скреча могу веома удобно наставити рад и у програмском окружењу BYOB. Иначе, BYOB је акроним поруке Build Your Own Blocks, односно позива да ђаци наставе да програмирају тако што сами осмишљавају програмерске модуле путем изградње програмских блокова. У језику BYOB је проширен концепт листи, процедура и ликова тако што сваки лик може да има сопствени блок и сопствени скрипт, док више ликова могу конкурентно да извршавају своје аутономне акције.

Занимљиво да иако су изражајна средства BYOB-а и SNAP-а богатија, ученици и даље имају могућност да веома рано виде резултат рада свог програма у почетној фази рада и тиме могу пажљиво да осмишљавају остале кораке моделовања и пројектовања програма. И даље је, као и у Скречу, тешко направити синтаксно некоректан програм што свакако доприноси идеји да се концепти програмирања и алгоритмике могу веома рано увести у систем образовања.

2. Ограничени конструкти СКРЕЧА и имплементација напредних техника програмирања у BYOB-у и SNAP-у

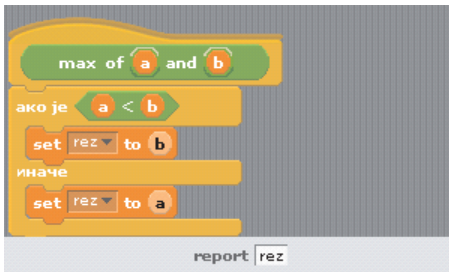
Скреч је наследник Logo језика, развијеног у MIT Media Lab и намењен је, пре свега деци. Али, сведоци смо растућег тренда на универзитетима да се на курсевима рачунарства за неинформатичаре и нематематичаре користи Скреч као програмерско окружење, због своје једноставности. Такође, визуелна употреба вишенитног програмирања у Скречу пружа једноставан увод у свет дистрибуираног израчунавања и паралелног програмирања. Један од таквих курсева је извођен као пилот пројекат на Универзитету у Калифорнији, Berkeley. И сам назив курса „Лепота и радост рачунарства“ (The Beauty and Joy of Computing) указује да се плану и програму курса приступало са посебном пажњом, као и да су се у реализацију укључили ентузијасти са вишегодишњим искуством у настави и програмирању. Интересантно је да су ђаци и студенти који нису на информатичким смеровима веома често бирали овај курс у списку изборних предмета.

И баш су на Berkeley-у уочили да Скречу као програмском језику недостају одређени конструкти и да му је потребно проширење како би могао да покрије наставу у средњим школама и на факултетима. На првом месту, недостају му процедуре, тако да не може да се користи у уводним курсевима програмирања који покривају важан феномен рекурзије, једне од централних идеја рачунарства (и, такође, централне идеје методологије програмирања у ранијем узрасту, тзв. Logo педагогије). Такође, слабо су подржане и структуре података. Ове слабости нису пропуст језика или превид његових твораца, јер пројектанти Скреча су свесно и промишљено избегавали да у језик додају све функционалне концепте програмирања који су деци тешки за савладавање или уносе пометњу и збрку код деце приликом учења програмирања. На другој страни, Brian Harvey, коаутор SNAP-а и BYOB-а у свом раду [6] истиче да Скреч може уз одређена проширења постати „програмски језик без плафона“ тј. постати подједнако користан и довољно функционалан и за децу и за научнике.

Дакле, да би Скреч могао да задовољи потребе две различите заједнице (деца и студенти), предложено је да Скреч заједница развија две верзије језика: верзију за децу и верзију за напредне кориснике. Brian Harvey верује да то није неопходно. Ако се преузме кључна идеја из Scheme (програмског језика функционалне парадигме програмирања) да су функције равноправне са свим другим типовима података и да можемо да користимо функције вишег реда, онда можемо додати само неколико својстава Скречу и ипак имати довољно моћно програмерско окружење као основу за озбиљан уводни курс у рачунарство и информатику. Подсетимо се да су функције вишег реда оне функције које могу да узимају друге

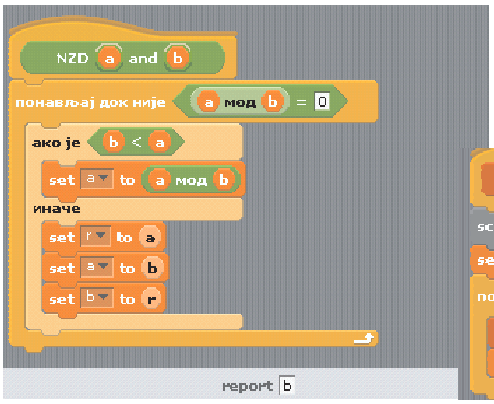
функције као своје аргументе, и да их враћају као резултат. Примери су оператори у математици, попут диференцијалног оператора d/dx који даје извод када се примени на функцију f. Проширење Скреча које може да задовољи основна својства функционалне парадигме програмирања је најпре представљено 2010. године у развојном окружењу BYOB, док актуелна верзија 4.0 окружења SNAP подржава и концепт ламбда рачуна. Штавише, графички интерфејс BYOB-а и SNAP-а доприноси поједностављењу имплементације процедура, јер подаци се чине мање апстрактни и мање плаше почетнике у програмирању. Приметно је да су задржане и неке од основних предности функционалног програмирања: програми су концизни и јасни, имплементирана решења су често краћа од решења имплементираних у процедуралној парадигми [5]. Теже је направити грешке, а ако постоје, лакше их је открити. Аутори BYOB-а и SNAP-а улажу напор и да се омогуће релативно једноставни докази коректности програма.

Сл. 1



Сл. 2

Пример 1. Програм који користи (блок) потпрограм за одређивање максимума два броја (сл. 1).



Сл. 3

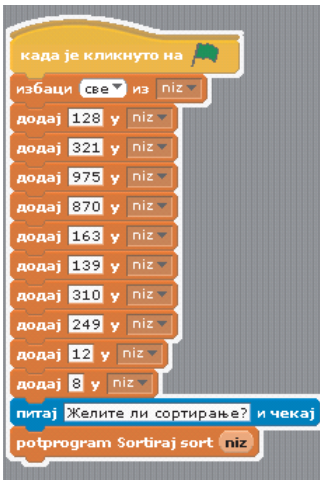
Сл. 4

Пример 2. Блок за одређивање максимума два броја a и b који се користи у примеру 1 (сл. 2).

Пример 3. Имплементација (блока) потпрограма за одређивање највећег заједничког делиоца два природна броја a и b (сл. 3).

Пример 4. Имплементација потпрограма за сортирање методом уметања (insertion sort) (сл. 4).

Ово је пример у ком је имплементирана процедура Sortiraj која као аргументе добија елементе низа смештене у структури листе. Потпрограм као повратну вредност враћа листу сортираних елемената.



Сл. 5

Пример 5. Главни програм који учитава чланове низа целих бројева и позива потпрограм за сортирање.

Програмски фрагмент *Pomeri se na poziciju* представља посебан блок који као аргумент добија елемент низа (листе) на позицији *index* и трампи тај елемент са следећим елементом. Потребно је да програмски блок понавља овај корак све док број не доспе на своју одговарајућу позицију у листи (врх листе или позиције испод мањег броја). На тај начин број на позицији *index* смештен је на своју тачну позицију у сортираном делу листе (проверите непосредно!).

3. Алгоритмика (BYOB и SNAP)

Програмерски курсеви у средњим школама и на универзитетима (а понегде и у основним школама) захтевају напредну алгоритмизацију формулисаних задатака у смислу да користимо што ефикасније алгоритме и што напредније структуре података. За то нам је потребно више конструката који не постоје у Скреч језику, те се у сврху изучавања напредних алгоритамских техника може користити BYOB и SNAP.

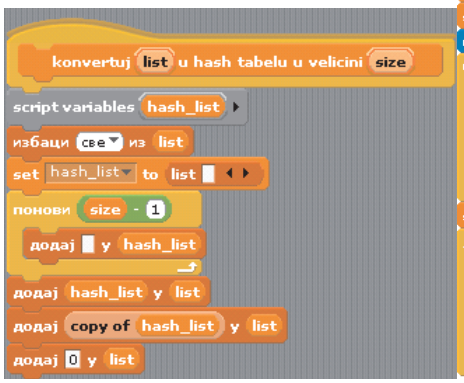
Пример 6. Илуструје техику брзе хеш претраге (за унету вредност кључа, добити број телефона из хеш табеле) (сл. 6).

Пример 7. Формирање хеш табеле за дате улазе из примера 6 (сл. 7).

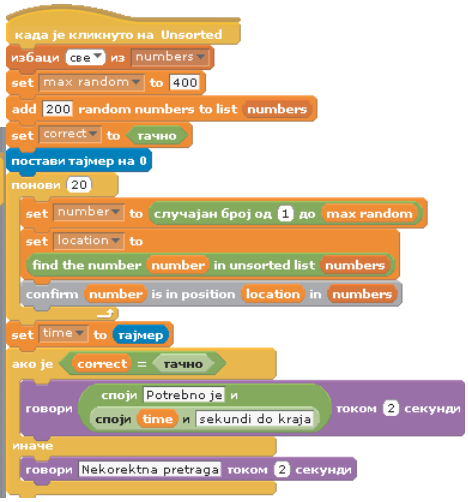
Пример 8. Генерисање случајног несортираног низа и претрага елемента у низу (сл. 8).

Пример 9 (општинско такмичење 2013, 2. категорија основне школе). Сваке године ученици имају прилику да у школским свечаним салама присуствују прослави школске славе. Разредне старешине су одлучиле да распореде ученике према прозивнику на следећи начин: најпре се у сали попуњава први ред у смеру слево надесно, потом на исти начин се попуњава други ред и тако редом док се не попуни цела сала која има н редова са по м места за седење у сваком реду. Међутим, кад је дошао директор школе, образложио је разредним старешинама

Сл. 6



Сл. 7



Сл. 8

зашто такав распоред седења није коректан и предложио је прерасподелу седења на следећи начин: у сваком реду (од првог до последњег) ученици треба најпре да попуне сва прва места, потом сва друга места и тако даље. Напишите програм PROSLAVA, којим се уносе цели бројеви n, m ($1 < m < 3 * 10^4$, $1 < n < 3 * 10^4$) и који израчунава и исписује колико ученика у прерасподели ће остати на својим првобитним местима. Сматрајте да школа има довољно ученика да попуни целу салу.

Тест пример 1.

Улаз	Излаз
3 3	3

Тест пример 2.

Улаз	Излаз
2 4	2

Објашњење за први тест пример: ако $n = 3$ и $m = 3$, првобитни распоред ученика је

1 2 3
4 5 6
7 8 9

Након прерасподеле, распоред ученика је

1 4 7
2 5 8
3 6 9

Ученици с редним бројевима 1, 5 и 9 ће остати на својим местима.

Овај задатак је спадао у најтеже задатке на Општинском такмичењу узимајући у обзир просечан број поена које су ученици освојили на овом задатку.

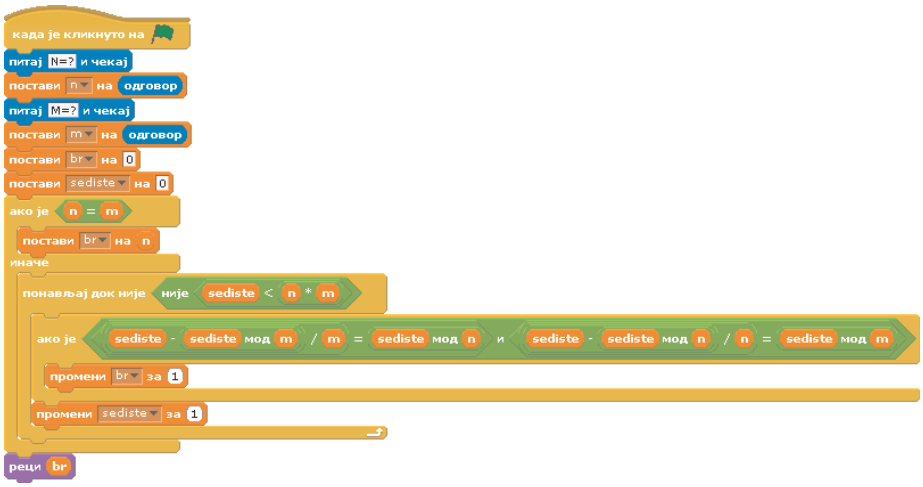
Анализа проблема: Седиште се не мења ако је редни број колоне расподеле једнак редном броју колоне прерасподеле и ако аналогно важи за врсте.

Следи решење у виду програма имплементираног у програмском језику C++.

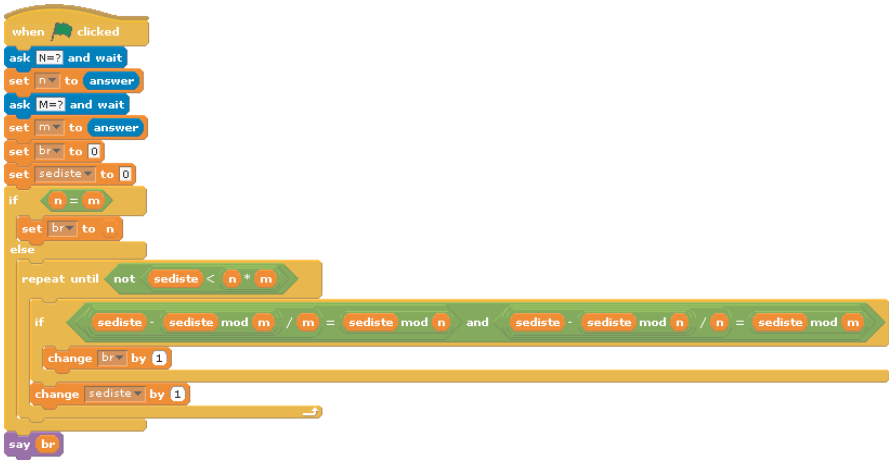
```
# include <iostream>
using namespace std;
int main()
{
    long int n, m; // vrsta, kolona
    cin>>n>>m;
    long int br = 0;
if (n==m) br=n;
    else
    for (int sediste=0;sediste<n*m;sediste++)
        if (sediste / n == sediste % m && sediste / m == sediste % n)   ++br;
          //sediste se ne menja ako je redni broj kolone raspodele jednak rednom
          //broju kolone preraspodele i ako analogno vazi za vrste
    cout<<br<<endl;
    return 0;
}
```

Дакле, одговарајуће решење у BYOB-у би се креирало користећи конструкте који постоје и у Скречу (сл. 9).

Сличан пример је могао бити решен користећи и интерфејс програма BYOB и Скреч који није локализован на српски језик (сл. 10).



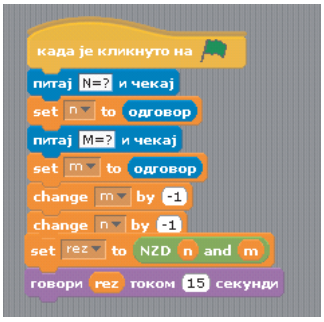
Сл. 9



Сл. 10

Али, ако задатак желимо да решимо користећи идеју да је довољно наћи највећи заједнички делилац за димензије подматрице са $n-1$ врстом и $m-1$ колоном, онда можемо имплементирати ефикасније решење уз компактнији код (в. наредну страницу).

Грешке такмичара. У овом задатку није било потребно користити низове (ни једнодимензионе ни дводимензионе низове). Неки такмичари су алоцирали превише меморије декларишући матрицу димензије $30000 * 30000$ и услед тога им програм није радио. Такође, нека решења грубом силом нису могла да дају коректан резултат у времену од 10 секунди. Било је и такмичара који су добили утешне поене на тест примерима за квадратну матрицу, тј. за случај $n = m$.



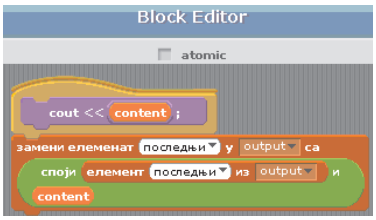
Сл. 11

```
# include <iostream>
using namespace std;
int nzd(int a,int b)
{
    while (b)
    {     int r = a % b;
        a = b;
        b = r;
    }
    return a;
}
int main()
{     int n, m;
  cin>>n>>m;
  cout<<1+nzd(n - 1, m - 1) << endl;
  return 0;
}
```

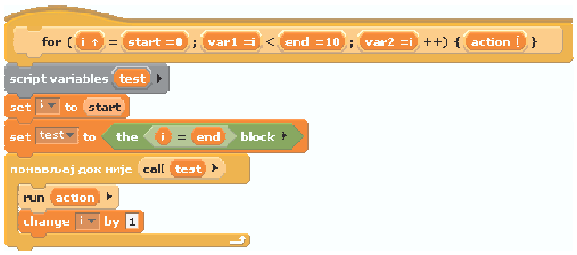
4. Алгоритмика у BYOB као увод за програмирање у вишим програмским језицима

Млади програмери могу релативно удобно напредовати са BYOB језика и ка другим програмским језицима. Релативно једноставно се прелази на програмске језике функционалне парадигме (као што су Haskell, F#, ...) с обзиром да BYOB третира програмске конструкције као наредбе за израчунавање математичких функција. Међутим, коришћењем BYOB може се направити и увод у програмирање на језику C/C++, као што се може у вишим разредима средње школе и на факултету изучавати теорија формалних језика или превођења програма тако што би се примитиве вишег програмског језика имплементирали као блокови или предикати у BYOB.

Пример 10. Имплементација конструката језика C++ у BYOB-у (cout блок) (сл. 12).



Сл. 12



Сл. 13

Пример 11. Имплементација конструката језика C++ у BYOB-у (for блок) (сл. 13).

Дакле, BYOB је лако проширив језик у смислу да можете сами осмислити примитиве ниског нивоа. То је важно својство које омогућује да се на факултетима са уводног курса програмирања може започети и изучавање напредних техника програмирања у складу са интересовањима студената.

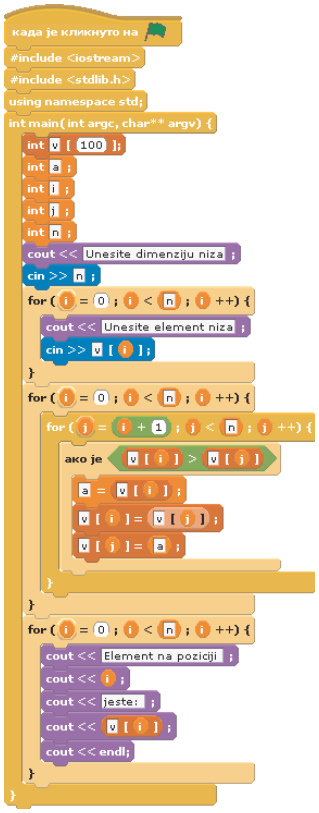
Пример 12. Имплементација програма за сортирање низова методом мехурића (bubble sort) (сл. 14).

Пример 13. Имплементација конструката језика C++ у BYOB-у (cin блок) (сл. 15).

5. Закључак и будући рад

Изучавање рекурзије је важно за сваки озбиљнији курс програмирања. Имајући на уму средњошколце и студенте као учеснике у образовном процесу, Jens Mönig је развио екстензију Скреча названу BYOB (Build Your Own Blocks) која омогућује употребу рекурзије једноставним креирањем нове Скреч процедуре.

Развојно окружење BYOB је доступно у online и offline верзији. Online верзија је креирана у Javascript-у и довољно је поседовати Internet прегледач (browser) да би се могао имплементирати и покренути BYOB програм. Аутори BYOB-а су сматрали да ће на тај начин BYOB бити приступачнији информатички неискуснијим корисницима с обзиром да не морају да обаве инсталацију на свом рачунару. Такође, актуелна online верзија може да се извршава на Apple iOS, Windows и Linux уређајима, захваљујући имплементацији у Javascript. Нажалост, актуелна верзија развојног окружења Скреч 2.0 је имплементирана у Flash-у и не може да се извршава на Apple iOS уређајима.

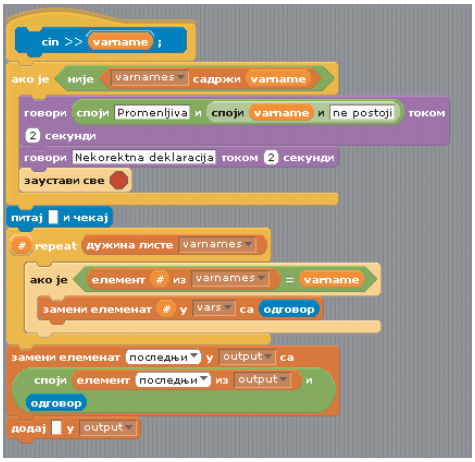


Сл. 14

Offline верзија је доступна на Windows, Mac OS X, Linux платформи и доступна је по правилима лиценцирања софтвера отвореног кода директно са Веб странице Универзитета Беркли.

Почев од 2012. године, постоји активна и све бројнија корисничка заједница SNAP-а и BYOB-а која на порталу `http://byob.berkeley.edu/` објављује у оквиру галерије пројеката своје програмерске радове, као и методичка искуства преточена у мале приручнике за примену SNAP-а и BYOB-а унутар учионица у оквиру разноврсних предмета (информатика, математика, биологија, учење страног језика, ...).

Сведоци смо да је Србија, иако је по површини релативно мала земља, успела да образује и подари значајним (светским) развојним тимовима много програмера. Велики број ученика, њихових родитеља и наставника препознаје потенцијал



Сл. 15

који нуди програмирање, али да би се неко бавио програмирањем мора да усвоји посебан начин размишљања и решавања проблема. Већина наставника информатике уочава да све веће присуство рачунара у друштву и школама, помаже да се деца при првом сусрету са рачунарима у образовном систему осећају опуштеније и да лакше савладају основе коришћења рачунара.

Али, нажалост, доступност рачунара није допринела олакшаном савладавању техника програмирања. Наиме, савремене технологије су пријатељски усмерене ка кориснику и створиле су код ученика став да све проблеме могу да решавају „дуплим кликом“, а све мањи број ученика улаже напор да се посвети решавању задатака посредством програмирања. Додатно, планови и програми наставе рачунарства и информатике у основним и средњим школама су прилагођени идеји да је потребно да се деца обуче за рад на рачунару, а све мање пажње се посве?ује настави програмирања [7]. Посебан проблем представља недостатак адекватне литературе, која би обухватила програмске језике, који су довољно моћни да испрате основне концепте алгоритмике, али и да не буду превише компликовани ученицима за савладавање.

Ово је први део текста о програмском језику и окружењу BYOB/SNAP који уз приложене примере говори о могућностима и општим карактеристикама језика и интерфејсу окружења. Позивамо све наставнике да нам изложе сопствена искуства у коришћењу BYOB/SNAP алата. У наредним наставцима детаљније ћемо објаснити рад са нискама знакова, листама и напредним структурама података које су ефикасне за операцију претраге. Група за локализацију Скреча при Активу информатике Математичке гимназије ради и на локализацији BYOB-а и до сада смо делимично извршили локализацију кључних језичких конструката на српски језик и ћирилицу, док смо одређене сегменте оставили у оригиналу на енглеском језику.

Задаци за самостални рад

1. Фабрика колача *Шумадија* је својим тортама освојила Европу. У фабрици су се специјализовали и за израду алгебарских торти које су специфичне јер садрже на свом врху чоколадни број. Низ чоколадних целих бројева се формира на следећи начин: сваки наредни број се добија из претходног броја коме је додат број његових делилаца. Нека a $(0 < a \leqslant 10000)$ означава први чоколадни број у низу. На пример, ако је $a = 6$, онда следећи члан низа је $6 + 4 = 10$, јер број 6 има 4 делиоца – 1, 2, 3, 6. Написати програм који учитава два цела броја (први чоколадни број у низу и цео број N) и исписује N-ти чоколадни број низа, $0 < N \leqslant 100$.

   Пример.

Улаз	Излаз
7 3	12

2. У Нишу је изграђена нова кружна тркачка стаза дужине x метара за обуку возача ауто-мото спортских клубова. Пера и Васа тренирају заједно на тој тркачкој стази. Обука обухвата неколико фаза, а свака се састоји од узастопних деоница на којима се аутомобили крећу различитом брзином. На почетку обуке, возачи су добили план који изгледа као следећа табела:

	Пера		Васа	
Бр.	брзина (m/min)	време (min)	брзина (m/min)	време (min)
1	−600	15	200	13
2	700	13	−250	33
3	−550	11	150	21

   Сваки ред садржи брзину (у метрима по минути) и време у минутима, које морају задржати два возача током одговарајуће фазе обуке. Напишите програм који израчунава минимално растојање у метрима (мерено на тркачкој стази) између два возача на крају трке, ако возачи стартују истовремено из једне тачке и не заустављају се ради одмора. Програм треба најпре да учита два цела броја x, N $(100 \leqslant x \leqslant 1000,\ 1 \leqslant N \leqslant 20)$ који редом означавају дужину стазе у метрима и број редова плана који су возачи добили на почетку трке. Потом треба учитати следећих N четворки, тј. четири цела броја: $v_1\ t_1\ v_2\ t_2$, тако да v_1 је Перина брзина; t_1 је време током ког Пера мора задржати ту брзину; v_2 је Васина брзина, t_2 је време током ког Васа мора задржати ту брзину. Претпоставити да је $t_1, t_2 > 0$, $-1000 \leqslant v_1, v_2 \leqslant 1000$. Ако је брзина већа од 0, аутомобил се креће у смеру кретања казаљке на сату. Ако је брзина мања од 0, аутомобил се креће у смеру обратном од смера кретања казаљке на сату.

Пример.

Улаз	Излаз
950 3 −600 15 200 13 700 13 −250 33 −550 11 150 21	350

ЛИТЕРАТУРА

1. Resnick, M., Maloney, J., Monroy-Hernandez, A., Rusk, N., Eastmond, E., Brennan, K., Millner, A., Rosenbaum, E., Silver, J., Silverman, B., and Kafai, Y., *Scratch: Programming for All*, Communications of the ACM, **52**, 11 (2009), 60–67.
2. Malan, D., and Leitner, H., *Scratch for Budding Computer Scientists*, SIGCSE Procedings, 2007.
3. Wolz, U., Maloney, J., and Pulimood, S.J., *"Scratch" Your Way to Introductory CS*, ACM SIGCSE Bulletin, **40**, 1, March 2008.
4. Shafer, R., *Oh! The Beauty and Joy of Computing*, Innovations **3**, 10, UC Berkeley College of Engineering, December 2009. (`http://inst.eecs.berkeley.edu/~cs10`)
5. Abelson, H., and Sussman, G.J., with Sussman, J., *Structure and Interpretation of Computer Programs*, 2nd edition, MIT Press, 1996.
6. Brian Harvey, Jens Mönig, *Bringing 'No Ceiling' to Scratch: Can One Language Serve Kids and Computer Scientists?*, Proceedings of Constructionism 2010 conference, Paris, France.
7. Милан Чабаркапа, Станка Матковић, Татјана Тимотијевић, *Збирка задатака из програмирања, окружна и републичка такмичења ученика основних школа 1988–2006*, ЦЕТ Београд, Друштво математичара Србије, Београд, Рачунарска гимназија, Београд, 2007.

Катедра за рачунарство и информатику, Математички факултет, Универзитет у Београду

E-mail: `jelenagr@matf.bg.ac.rs`