Deși întrebările parametrizate pot fi utile pentru a preveni plagiatul, ele necesită destul de mult timp din partea cadrelor didactice pentru crearea tiparelor. Un potențial răspuns la această problemă este furnizat de lucrarea ce prezintă o abordare pentru generarea procedurală a exercițiilor de programare fără a utiliza șabloane parametrizate. Pentru a reprezenta exercițiile într-o manieră structurată, fiecare exercițiu este tratat ca o succesiune de instrucțiuni pe care elevul trebuie să le efectueze. Fiecare instrucțiune este reprezentată de un singur nod, care poate fi asociat cu o operație, o condiție sau returnarea unei valori. Prin urmare, exercițiul este o colecție de noduri, care va fi parcursă succesiv.

Un nod de operație, se referă la o singură operație aritmetică și este urmat întotdeauna de un singur nod. Un nod de condiție, se referă la o singură expresie condițională și este întotdeauna urmat de două noduri, unul pentru când condiția este adevărată și unul pentru când condiția este falsă. Un nod de returnare, se referă la un punct final al funcției și nu este urmat de un alt nod. Operații de nivel superior pot fi adăugate prin abstractizarea lor ca blocuri de operații sau de condiții. Blocurile constau într-o secvență internă de operații elementare.

Nodurile de operație au forma „C = A op B”, unde A și B sunt operanzii, op este un operator aritmetic de bază, cum ar fi adunarea (+) sau înmulțirea (\*), iar C este variabila de ieșire unde este stocat rezultatul. Nodurile de condiție au forma „A op B”, unde A și B sunt operanzi și op este un operator condițional de bază, cum ar fi mai mare decât (>) și mai mic decât (<). Nodurile de returnare au forma „return A”, unde A este valoarea care trebuie returnată.

Pornind de la o valoare a complexității, se generează o structură aleatorie. Complexitatea unei structuri este o metrică definită ca numărul de noduri asociate cu o operație sau cu o condiție. Această definiție se bazează pe ideea că un exercițiu devine mai complex pe măsură ce crește numărul de instrucțiuni necesare pentru a-l finaliza.

Primul nod este inițializat fie ca nod de operație fie ca nod de condiție. Apoi, într-unul din locurile disponibile din cadrul structurii este atașat aleatoriu, în mod asemănător, fie un nod de operare, fie un nod de condiție. Această operațiune este efectuată până când structura conține un număr de noduri egal cu complexitatea. În locurile libere rămase, sunt atașate nodurile de returnare.

După definirea structurii exercițiului, parametrii fiecărui nod sunt asignați utilizând un algoritm ce a luat în calcul considerațiile detaliate în cadrul lucrării. Adâncimea unui nod N este definită ca numărul de pași necesari pentru a ajunge la nodul N de la nodul de început, iar Clist este definită ca o listă de noduri critice. Este considerat un nod critic, un nod a cărui operație influențează rezultatul funcției. Inițial, din Clist fac parte doar nodurile de returnare și nodurile de condiție. Nodurile de returnare influențează direct valoarea de retur a funcției prin definiție, în timp ce nodurile de condiție au un rol important în alegerea unei anumite ramuri, influențând astfel ce nod de returnare va fi atins.

Algoritmul utilizează o variabilă contor ce pornește de la indexul 1. Fiecare nod de operație este parcurs în ordinea descendentă a adâncimii. Pentru fiecare nod de operație N, variabila Xi este atribuită ca variabilă de ieșire, contorul este incrementat, și un nod C este selectat aleator din Clist. Nodul C este testat dacă îndeplinește următoarele condiții: are cel puțin un operand care nu este încă definit și este accesibil din N. Dacă niciun nod din listă nu îndeplinește condițiile de mai sus, generarea nu este considerată validă. În caz contrar, variabila Xi este atribuită unuia dintre operanzii lui C. Deoarece C este un nod critic și variabila de ieșire a lui N este acum utilizată ca operand al lui C, rezultă că N este acum un nod critic, deci trebuie adăugat în Clist. În mod aleatoriu, algoritmul poate repeta atribuirea Xi la mai multe noduri din Clist, dar este necesară doar prima atribuire. Acest proces se repetă până când toate nodurile se află în Clist. Fiecărui nod care nu are încă operanzi, îi sunt atribuite cel puțin una din variabilele I1, I2, I3 ca operand. Aceste variabile reprezintă parametrii transmiși funcției asociate exercițiului. În cele din urmă, operanzilor care nu au fost definiți, le sunt atribuite valori constante aleatorii.

Odată construită secvența de instrucțiuni, fiecare nod este parcurs și în funcție de tipul acestuia este generat atât un text în limbaj natural pentru crearea exercițiului, cât și codul Java asociat pentru a evalua soluția elevului. Există mai multe texte predefinite pentru fiecare tip de nod, iar din acestea este ales unul, în mod aleator. Ulterior, porțiunile marcate corespunzător sunt înlocuite cu conținutul variabilelor din cadrul structurii nodului.

[ce nu e bun]

Cu toate acestea, mai e loc de îmbunătățire în abordarea noastră. În acest studiu, am folosit doar operații simple de calcul pentru generarea exercițiilor de programare. Aceasta include operațiile aritmetice (de exemplu, adunarea, înmulțirea), operațiile condiționale (de exemplu, mai mare decât, mai mic decât) și câteva blocuri, cum ar fi obținerea valorii absolute și obținerea valorii mai mari între două numere întregi. De asemenea, ne-am limitat tipurile de date la numere întregi. Din acest motiv, exercițiile generate de sistemul nostru, în ciuda faptului că nu sunt derivate dintr-un șablon parametrizat, arată în continuare similar între ele. Abordarea noastră actuală este, de asemenea, constrânsă de operații matematice independente, abstracte. Se presupune că fiecare operație nu este dependentă de niciuna dintre operațiile anterioare și nu se presupune niciun context pentru fiecare operand în afară de faptul că sunt numere întregi. Acest lucru limitează sever cât de interesante sunt exercițiilor generate, deoarece operațiunile par niște calcule disjuncte care nu formează o poveste semnificativă. Credem că abordarea noastră actuală pentru reprezentarea exercițiilor poate servi drept fundație pentru generarea de exerciții stabilite în contextul lumii reale. Prin maparea acțiunilor din lumea reală, cum ar fi conversia metrilor în picioare sau calcularea modificării unei secvențe de operații, ar putea fi posibilă utilizarea tehnicilor de generare a diagramelor pentru a genera exerciții de programare care conțin nu numai operații matematice abstracte, ci și aplicații din viața reală. programare. Făcând acest lucru, putem produce exerciții care conțin operații care sunt coerente într-un context real, ceea ce le poate face mai interesante de rezolvat de către elevi.