Pentru a stăpâni limbaje de programare nu este de ajuns doar înțelegerea gramaticii limbajului și a instrumentelor de programare. Este indispensabilă efectuarea în mod repetat a multor exerciții practice de programare.

[recent works]

s-au depus mult efort în dezvoltarea resurselor și sistemelor de învățare a programării. Cu toate acestea, sistemele și resursele actuale pentru învățarea programarii se bazează pe conținut creat manual, fapt ce devalorizează sistemul, prin numărul limitat de redări

Munca privind generarea automată de exerciții pentru programare a fost relativ limitată în comparație cu alte domenii. Studiile anterioare au arătat tehnici pentru generarea de probleme în algebră [4] și geometrie [5]. În programare, generarea automată a exercițiilor a fost limitată la utilizarea întrebărilor parametrizate, în care variațiile problemei sunt derivate din un șablon de exercițiu. În lucrarea lui ... și colab., cuvintele sau frazele sunt înlocuite în problemele de programare pentru a produce diferite variații. În lucrările lui ... și ..., șabloanele de program C sunt folosite ca bază pentru generarea de exerciții de urmărire și depanare. În lucrările lui ..., ...., ..., programele Java parametrizate orientate pe obiecte sunt utilizate pentru urmărirea întrebărilor [8].

În această lucrare ne concentrăm pe aspectul generării exercițiilor, care permite sistemului să prezinte conținut dinamic sub formă de exerciții de codare. Definim un exercițiu de codificare ca un exercițiu în care studentul scrie cod care face ceva pe baza unei specificații date. Un exemplu este să scrieți un program care adună două numere împreună.

[intro design]

Discutăm o abordare în generarea procedurală a exercițiilor de codare pentru practica de programare fără a utiliza întrebări parametrizate.

În această lucrare, noi abordează două probleme principale. În primul rând, cum reprezentăm exercițiile de codificare într-o manieră structurată? Și în al doilea rând, cum generăm exerciții automat folosind acea reprezentare?

Sistemul oferă elevului o interfață pentru a scrie codul, a testa codul furnizând valori eșantion și a trimite codul pentru verificare. Dacă soluția este corectă, sistemul generează următorul exercițiu. Dacă un student nu poate rezolva un exercițiu timp de șapte minute, devine disponibilă o opțiune de a renunța și de a obține un alt exercițiu

[design]

Pentru acest studiu, am folosit o metrică de bază pentru complexitate, care este numărul de noduri de operare și condiție din exercițiu. Aceasta se bazează pe ideea că un exercițiu devine mai complex pe măsură ce crește numărul de operațiuni necesare pentru a-l finaliza.

Am folosit Java ca limbaj de programare, deoarece este folosit în mod obișnuit în clasele introductive de programare. În sistem, elevul rezolvă o serie de exerciții în care corpul unei funcții trebuie finalizat pentru a îndeplini o sarcină specificată.

[la finalul descrierii sistemului]

Trebuie remarcat faptul că această lucrare prezintă cazul limbajului C, dar abordarea noastră poate fi extinsă cu ușurință la alte limbaje de programare. Sistemul nostru prototip este situat la <http://pplinux.is.konan-u.ac.jp/rccE.html>.

Prin urmare, deoarece cursanții nu vor observa că efectuează în mod repetat programe generate din același șablon, sistemul nostru este potrivit pentru învățarea repetitivă. Trebuie remarcat faptul că programele șablon sunt pregătite în conformitate cu progresul învățării și sunt necesare aproximativ 2, 3 de programe șablon pentru a face față unei varietăți de programe.

Trebuie remarcat faptul că abordarea noastră cu același script PHP este ușor extinsă la alte limbaje de programare prin schimbarea compilatoarelor și rescrierea programelor șablon și a matricelor sp și rp.

După cum s-a menționat mai sus, parametrii constanti și modelele de calcul ale programelor trebuie să fie date corect, deoarece parametrii greșiți pot provoca excepții de diviziune la zero și erori de acces la memorie.

[rezultate experimentale]

De asemenea, confirmăm eficacitatea sistemului prin experimentele noastre preliminare.

eficacitatea practicilor de depanare este confirmată empiric. În viitor, vom evalua următoarele aspecte: • evaluarea practicilor de estimare a rezultatelor, • combinația dintre sistemul nostru și MOOC, • exercițiile în conformitate cu progresul învățării și • evaluarea profundă de către mulți evaluatori. Mai mult, ar trebui să implementăm și sistemul practic și să-l evaluăm

[concluzii]

Testarea noastră arată că exercițiile de programare generate procedural sunt promițătoare pentru a ajuta elevii să învețe sau să exerseze codificarea.

Această abordare nu numai că poate genera dinamic exerciții, dar poate, de asemenea, să verifice în mod automat trimiterile studenților. Cu toate acestea, există încă multe îmbunătățiri de făcut în procesul de generare a exercițiilor pentru a putea crea probleme mai interesante. Munca viitoare include integrarea cunoștințelor de bun simț în procesul de generare a exercițiilor pentru a crea mai semnificative exerciții care se bazează pe aplicații din lumea reală, precum și o analiză mai empirică a momentului în care să se furnizeze intervenții și ce tipuri de personalizări sunt adecvate pentru diferite tipuri de scenarii de confuzie.