**[descrie sistemul] -> [am inceput cu ceva simplu], [deep dive into grammar | garbage, exploration], [rezultatul]**

Sa zic ca am ales c++ pentru ca aveam portiunile de cod din cartea aia de teste

Imagine cu codul generat si AST-ul corespunzator

Pe langa conceptul de gramatica am adoptat si alte concepte din lumea compilatoarelor

Această lucrare prezintă o abordare generativă în rezolvarea acestor probleme. Este introdus sistemul de generare automată a exercițiilor care sunt create după o gramatică și sunt unice pentru fiecare student. Această abordare, aplicată în timpul procesului de predare, are scopul de a reduce dramatic probabilitatea ca soluțiile să fie copiate, iar ulterior are ca rezultat o implicare mai activă a fiecărui elev pentru a crea o soluție unică și proprie. Rezultatul final este constituit de cunoștințe mai aplicabile în programare.

Generatorul on-line de exerciții ale elevilor se bazează pe un model de scriptare generator (GSM), introdus în ... de .... și descris de ... în ... în. Sistemul de generare implementat este prezentat în Figura 1.

**[arata ca poate fi folosit in alt domeniu, matematic], [fa comparatie ca e mai bun ca un produs de pe piata Brio]**

**[tehnologii folosite]**

Generatorul a fost incastrat in

Baza de date sql fiindca scopul este dezvoltarea unui mvp care sa demonstreze capabilitatile si utilizator putini

**[performanțe]**

Generatoarele de exerciții de programare discutate în această lucrare pot fi implementate în cadrul cursului de Structuri de Date și Algoritmi.

Tabelul prezintă performanța generatoarelor de exerciții de programare implementate în limbajul C#. Testul de performanță utilizează următoarea configurație: AMD Ryzen 7 4800U CPU 1.8 GHz, 16 GB RAM, C# 10.0, .NET 6.0. Rezultatele arată că schema propusă pentru construcția generatoarelor de exerciții de programare este destul de rapidă pentru a fi folosită online.

Tabel - Performanțele generatoarelor de exerciții de programare (tip exercițiu generat / performanța generator (exerciții/sec))

**[direcții]**

Utilizarea unui sistem automatizat în cadrul lecțiilor practice de programare ridică problema asigurării calității actului educațional produsă de lipsa componentei umane pentru verificarea detaliată a soluțiilor sau pentru personalizarea predării precum selecția individuală a complexității exercițiilor și obținerea de feedback detaliat asupra greșelilor.

Adaptarea dificultății exercițiului la nivelul unui anumit elev.

**[concluzii]**

Această lucrare a examinat problema scalării cursurilor de programare în contextul nevoii masive de specialiști a industriei tehnologiei informației și a sugerat utilizarea generatoarelor de exerciții ca una dintre abordările promițătoare pentru rezolvarea acestei probleme.

Este prezentată o schemă generală pentru construirea unui generator de exerciții de programare. Sunt identificate două clase principale de exerciții, ale căror variante pot fi automatizate: exerciții care necesită un algoritm specific pentru rezolvare sau exerciții care presupun citirea unei secvente de cod și deducerea output-ului. [more bla bla rezumat]

Experiența proprie a utilizării generatoarelor de exerciții de programare arată că exercițiile pot fi generate individual pentru fiecare student și sunt fezabile pentru mediul online.

Abordarea propusă în lucrare demonstrează că generatoarele de exerciții de programare pot produce o gamă largă de exerciții și necesită soluții netriviale din partea elevului, dezvoltându-i astfel abilitățile de programare.

[garbage]

Evaluarea automată a sarcinilor de programare este cel mai obișnuit exemplu de evaluare în domeniul informaticii. Cele mai vechi sisteme de evaluare, s-au bazat pe o metodă foarte simplă de potrivire a rezultatelor, unde este comparat rezultatul creat de un program model alcătuit de către profesor cu rezultatul programului elevului. Ulterior, acestea au evoluat și pe langă corectitudine, analizează și eficiența sau stilul de programare. Exemple mai recente includ utilizarea arborilor de sintaxă pentru a determina dacă o soluție este corectă din punct de vedere sintactic sau dacă studentul este capabil să urmeze convențiile de codare larg acceptate (să indenteze cod, să folosească comentarii etc.).

În [7], este descrisă o metodă de generare a exercițiilor introductive de programare. Generatorul construiește un program aleatoriu fără bucle, care constă în operații simple. Apoi, se generează un text în limbaj natural, care descrie succesiunea operațiilor din program. Sarcina elevului este de a traduce textul în limbajul natural în limbajul de programare ales. Dezavantajele acestui generator sunt următoarele:

• Un program aleatoriu care urmează să fie scris nu aparține niciunui domeniu aplicat, astfel încât studentul poate să nu fie interesat de soluție.

• Soluția este banală, fără a folosi abordările pe diferitele niveluri ale problemei.

Schema generală a unui generator de exerciții de programare

Acest articol prezintă o schemă generală de realizare a generatoarelor de exerciții de programare.

Aceste generatoare produc exerciții din diferite categorii, cum ar fi, liste, stiva

Utilizează o abordare

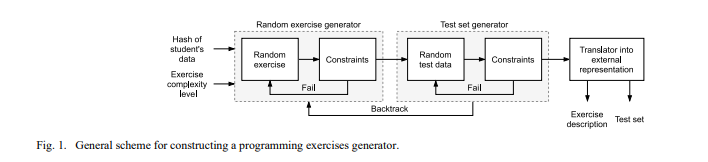
Utilizarea practică a metodelor descrise este prezentată în cadrul...

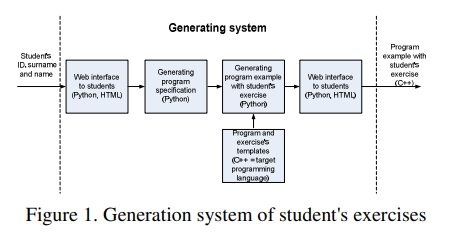
SCHEMA GENERALĂ A UNUI GENERATOR DE EXERCIȚII

În schema propusă în figura X, generarea exercițiilor conține următorii pași principali

1) crearea unui exercițiu

2) crearea ...





identifică o clasă de exerciții unde generarea poate fi automatizată

experiența utilizării unui astfel de generator.

Utilizarea generatoarelor de exerciții în procesul educațional nu este o noutate. Există, în special, generatoare de exerciții de algebră [1], geometrie [2, 3], logică matematică [4], teoria automatelor [5], precum și pentru cursul sistemelor integrate [6]. Mult mai puțină muncă este dedicată automatizării generării de exerciții de programare care necesită ca un student să evalueze o soluție.

Generatorul de exerciții aleatorii conține două module:

1) Crearea unui potențial exercițiu.

2) Verificarea rezultatului folosind un set de constrângeri.

Modularitatea simplifică implementarea și permite reutilizarea părților individuale ale generatorului de exerciții.

Este posibilă definirea următoarelor clase de exerciții generate

Dacă exercițiul aleator generat nu a satisfăcut setul de constrângeri, atunci acesta este eliminat.

Când se generează o suită de teste, rezultatul este verificat folosind constrângerile specificate, cum ar fi unicitatea cazurilor de testare sau numărul de cazuri de testare generate.

Dacă nu a fost posibil să se formeze un set de teste cu constrângerile specificate, atunci backtracking este utilizat la pornirea generatorului de exerciții aleatorii.

Dacă generarea suitei de testare a avut succes, reprezentarea internă a rezultatului este tradusă într-o formă externă, care poate include

Rezultatul generat folosind generatorul de exerciții aleatoare depinde de clasa de exerciții aleasă. Poate fi:

• Un program într-o reprezentare intermediară.

• O descriere a formatelor de date de intrare și de ieșire.

Pentru mulți generatori de exerciții care convertesc o notație în cod, este obișnuit să se construiască o expresie aleatorie ca arbore de sintaxă abstractă.

Este convenabil să descriem expresia pentru generare într-o formă executabilă asemănătoare BNF

În funcție de clasa exercițiilor, generatorul poate folosi

Generarea unei expresii poate produce recursivitate infintă. Modalitatea de rezolvare

Generatoarele de exerciții, vezi Fig. 3(a-c), sunt făcute pentru a antrena abilitățile de bază în lucrul cu condiționale, bucle și recursivitate.

Generarea începe cu

oferă fiecărui student un set individual de exerciții.