Raport Proiect 4

Computing Arithmetic Expressions

Descrierea codului

Proiectul a fost implementat in C++ folosind programare orientata pe obiecte.

Structura sablon `Node` :

- Poate fi instantiata cu orice tip de date numeric
- Are 4 campuri: *type*, *value*, *left*, *right*.
- `type` reprezinta tipul de informatie a nodului: *'none'* (tipul implicit folosit doar la initializare), *'number'* (numar) sau *'symbol'* (operatie aritmetica).
- `value` : daca tipul nodului este '*number*', atunci *value* este un tip de date numeric, iar daca tipul nodului este '*symbol*', atunci *value* este un caracter ASCII ce reprezinta o operatie aritmetica.
- `left` si `right` sunt pointeri la fiul stang, respectiv fiul drept al nodului.

Clasa sablon SyntaxTree:

`SyntaxTree` este un arbore binar, compus din noduri de tip `Node` , ce va retine expresia aritmetica data ca input in *prefix notation*. Clasa are doua functii publice necesare pentru rezolvarea problemei:

- `buildFromString(str)` ce primeste un c-string (`char*`) *str* reprezentand expresia aritmetica in forma prefix si construieste arborele binar corespunzator, parsand fiecare token din *str*. Pentru a retine ce nod trebuie sa fie construit la fiecare pas, se foloseste o stiva unde sunt adaugate noduri la fiecare paranteza deschisa intalnita.
- `evaluate()` , ce evalueaza expresia aritmetica retinuta in arbore. Ca principiu, functia foloseste o parcurgere in postordine pentru a afla valorile numerice asociate ambilor fii, apoi aplica operatia din radacina subarborelui si intoarce rezultatul.

In cadrul functiei `main` a fost exemplificata crearea si evaluarea unui arbore de sintaxa pentru tipul numeric `int` . Pentru convenienta folosirii programului se recomanda ca citirea sirului de caractere cu expresia aritmetica sa se faca din fisier, insa API-ul clasei nu impune acest lucru.

Restrictii ale programului

- Numerele din expresia aritmetica citita nu pot fi negative in implementarea curenta, insa acest lurcru poate fi rezolvat daca sunt adaugate mai multe conditii particulare in parsarea string-ului sau daca este folosita o librarie externa specializata pe parsari si extrageri de tokens.

- Din cauza ca parsarea trebuia facuta manual, am impus regula de a separa toti tokenii prin spatii, pentru a simplifica codul.

Exemple

- `( * ( + 3 5 ) ( - 7 4 ) )` => 24
- `( * ( - 3 1 ) ( + 2 5 ) )` => 14
- `( * ( / 10 ( - 5 3 ) ) ( + ( + 2 3) 5 ) )` => 50