Profesor: Anca Marginean
Student: Timotei Molcut
Grupa: 30239

Documentatie proiect
Grafix – interpretor pentru desenare de grafice

Descriere generala

Proiectul de fata reprezinta un interpretor pentru niste functii matematice de baza, iar in urma interpretarii cu succes se deseneaza graficul functiei respective. Aceste functii sunt de urmatoarele tipuri: polinoame, sinus/cosinus de polinom, si expoentiale de polinom. De asemenea, am introdus structuri precum for si if, acestea din urma influentand felul in care evolueza programul.

Motivatie

Am avut dorinta de a face acest interpretor in liceu (clasa a XII-a), pentru atestatul de informatica. Pe vremea aceea eram inca un incepator in domeniul informaticii si, prin urmare, am fost ajutat de catre profesorul meu de informatica pentru a face parsarea. Insa, astazi, avand cunostinte legate de Lex & Yacc, am reusit sa realizez parsarea doar cu ajutorul acestor tool-uri. Ceea ce vreau sa spun este ca am avut de mult timp pasiunea aceasta legata de grafice de functii matematice.

Implementare

Dupa cum am spus in paragraful de mai sus, am folosi Lex si Yacc. Lex este folosi pentru a captura expresii regulate dintr-un input de tip text, iar Yacc este folosit pentru a grupa aceste expresii regulate in reguli gramaticale care folosesc tokeni (token = element din limajul gramaticii). Astfel, Lex si Yacc pot fi folositi pentru a interpreta diferite tipuri de limbaje. In cazul de fata numele limajului este “Grafix” (deoarece deseneaza grafice cu variabila ‘x’).

Mai intai a trebuit sa recunosc toate tipurile de structuri din limbaj (for, if, polinom si structuri imbricate de acestea), iar la o recunoastere cu succes am creat un arbore de parsare. Prin aceasta, interpretorul capata viteza, deoarece mai intai incearca sa vada daca gaseste o structura corecta, iar la final o va executa. De aceea, la fiecare final de progam trebuie inserata comanda “draw”, care va executa fiecare arbore de parsare in parte si va deschide consola care deseneaza graficele.

Graficele recunoscute in acest moment de catre interpretor reprezinta o structura de polinom generala. Acest polinom poate fi un simplu monom (ex: 2x^3), o insiruire de monoame (ex: 2x-3 + x^5), sinus de polinom/cosinus de polinom (ex: sin(x) sau cos(3x-3)) si exponentiala de polinom (ex: 2^(x) sau 2^(x^2-x)). De asemenea, se pot face asignari la anumite variabile, iar apoi acele variabile pot fi folosite in grafice, precum in acest exemplu:

```
a = 2
x^a
draw
```

In urma executiei acestui exemplu se va desena parabola x^2. Daca nu se fac asignari la variabile, dar se folosesc in grafice, valoarea implicita a acestor variabile este zero.

Pentru a desena graficele am folosit functiile din biblioteca <graphics.h> (in special putPixel(x, y, color)). De asemenea, a trebuit sa fac o transformare de la gurpul de coordonate logice (x, y) la grupul de coordonate fizice din consola de desenare, adica pixel-ul propriu-zis. Pentru aceasta transformare am creat functiile getX si getY din fisierul Yacc. Pentru a reprezenta un

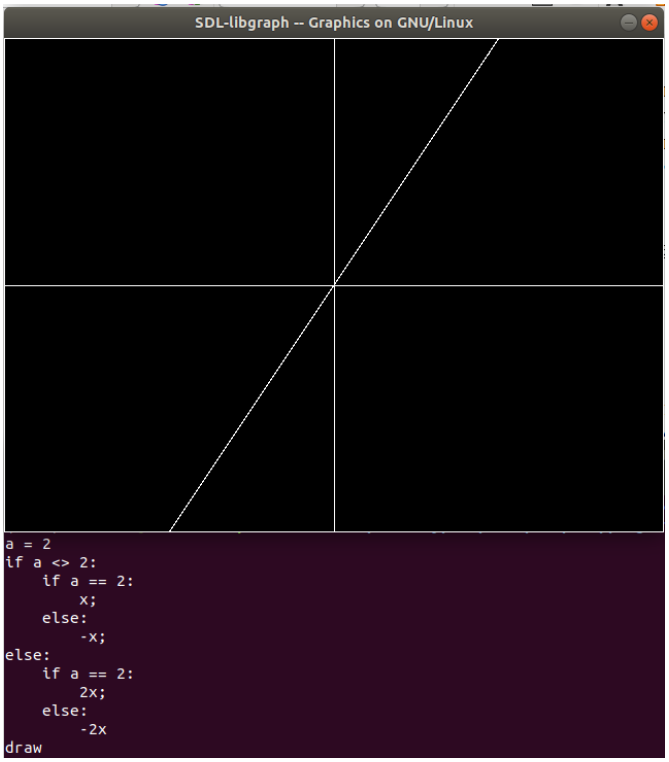
polinom am folosit un tablou de coeficienti de tip float, iar pentru evaluarea polinomului intr-un punct am creat functia valueOfPoly.

Dupa gasirea comenzii “draw” se vor evalua toti arborii de parsare cu functia handleTree, care in functie de tipul nodului din arbore va executa un cod sfecific case-ului din switch. Prin urmare am folosit un array de arbori de parsare pentru tot programul. Pentru variabile am folosit un array de valori float, iar pentru conditiile de “if” am facut un array de valori int (1 e true, iar -1 e false).

Rezultate experimentale

Exemplu cu for imbricat:

Exemplu cu if imbricat:



Concluzii

Interpretorul de fata se poate dezvolta mult mai mult astfel incat sa recunoasca structuri de tipul sin(…) + cos(…) + e^(…), dar in momentul de fata functioneaza corect doar cu functiile de baza, amintite mai sus.