SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

SEMINAR

Algebra u programskim jezicima

Davor Češljaš

Voditelj: Marijana Greblički

Zagreb, travanj, 2016

Sadržaj

1. Uvod

Moderniji računalni algebarski sustavi krenuli su se razvijati u drugoj polovici 20. stoljeća. Računalni algebarski sustav *(engl. Computer algebra system - CAS)* je alat koji omogućava da se matematički izrazi računaju preko računala. Svaki računalni algebarski sustav sastoji se od nekoliko dijelova:

- Korisničko sučelje
- **Programski jezik i njegov interpreter ili prevoditelj** *(engl. Compiler)*
- Dio programa koji pojednostavljuje formule *(engl. Simplifier)*
- Upravitelj memorije koji čisti sve nepotrebne međurezultate , a koji uključuje sakupljač smeća *(engl. Gargabe collector)*
- Velike biblioteke algoritama

U današnje vrijeme neki od poznatijih računalnih algebarskih sustava su zasigurno MatLab *(engl.Matrix Laboratory)* , Mathematica, odnosno njezina internet inačica WolphramAlpha te SymPy i ostali.

U ovom seminaru bavit ću se programskim jezicima koji se koriste kao podloge za računalne algebarske sustave. Dakako, takvih jezika ima iznimno mnogo te ću ovdje navesti samo programske jezike Phyton, C++ i Javu.

Na početku su navedene osnovne algebarske operacije.

Nakon njih objašnjeni su pojmovi razreda*(engl. class)* i biblioteke *(engl. library)*. U ostatku seminara upoznaje se sa sintaksom i različitim funkcionalnostima u svakom od navedenih programskih jezika.

Svaki jezik je detaljno potkrepljen službenom dokumentacijom za određenu klasu/biblioteku. Dakako, sve je potkrepljeno mnoštvom primjera.

2. Osnovne algebarske operacije u programskim jezicima

Pod ovim pojmom misli se na zbrajanje, oduzimanje, dijeljenje i množenje.

U arhitekturi samog procesora te operacije moguće je napraviti pomoću binarne aritmetike i operacija posmak, zbroj i rotacija. Pošto ih može direktno izvoditi sam procesor zovu se osnovne operacije.

a. Nekoliko primjera

U navedenim programskim jezicima sintaksa za osnovne algebarske operacije je uglavnom ista. Moguće su dvije forme (sljedeći kod biti će napisan u Javi i Phytonu):

1. Rezultat operacije i operandi su različite varijable:

```
//Java kod

…

int a = 5;

int b = 7;

int r;

r = a+b;

…

#Phyton kod

a = 5

b = 7

r = a/b
```

2. Rezultat i jedan operand su ista varijabla:

```
//Java kod

…

int a = 5;
```

```
int b = 7;

a += b;

…
```

```
#Phyton kod

a = 5

b = 7

b = a/b
```

Vidljivo je kako su ovo vrlo jednostavni kodovi i kako možda ne nalaze primjenu u računalnim algebarskim sustavima. Dakako, to nije točno. Zapitajmo se što nam treba za napraviti sumu geometrijskog reda danog formulom:

Jednadžba 2.1 Suma geometrijskog reda

$$S(x) = \sum_{n=0}^{\infty} x^n \tag{2.1}$$

```
#kod u Phytonu

sum = 0

steps = 100000

for n in range(steps):

      sum += pow((1/2),n)
```

Ovime se pokazuje da čak i osnovne algebarske operacije imaju svoje primjene u računalnim algebarskim sustavima te da se od njih mogu graditi i neki ne tako jednostavni algebarski izrazi.

Za još složenije formule ipak se treba upoznati sa funkcionalnostima matematičkih klasa/biblioteka.

3. Što su biblioteke i klase?

Prije no što se upoznamo sa matematičkim klasama i bibliotekama prvo ćemo se upoznati sa samim pojmom klasa i biblioteka.

a. Klasa

Mnoga bića, stvari i pojave nastojimo opisati na neki način. Svakom od njih pridaju se neki atributi i pokušava se opisati njihov rad, ponašanje i slično.

Podatke u objektno-orijentiranim jezicima dijelimo na podatke primitivnog tipa (int, char, float, double,…) i na objekte. U kontekstu računala i njegovih primitivnih tipova (kao što su brojke i slova) , teško se može opisati neki složeniji događaj ili opisati neko biće. Upravo se za takve stvari koriste objekti ,odnosno primjerci klasa.

Svaka klasa sastoji se od atributa , konstruktora i metoda nad tom klasom.

U atribute upisujemo vrijednosti koje označuju određeni objekt, a metodama mijenjamo neki atribut ili donosimo neku drugu funkcionalnost.

U konstruktoru stvaramo novi objekt u memoriji i opcionalno definiramo početne vrijednosti atributa. Konstruktor može i ne mora postojati. Ako konstruktor ne postoji to obično znači da su metode i atributi unutar klase statičke. Želimo li koristiti funkcionalnosti takve klase moramo je samo uvesti u program preko naredbe *import*.

Takav jedan razred je upravo razred Math u Javi koji ćemo detaljno razmatrati u slijedećem odlomku.

Za detaljniji opis savjetujem pogledati literaturu. [2]

b. Primjer klase

Kod kojim radimo zbrajanje dva broja u objektno-orijentiranom jeziku može izgledati i ovako:

```
public class Zbrajanje {
	//atributi
	private int a;
	private int b;

	//konstruktor
	public Zbrajanje(int a, int b) {
		this.a = a;
		this.b = b;
	}

	//metoda
	public int zbroji() {
		return a+b;
	}
}
```

c. Biblioteka

Biblioteka i razredi samo sa statičkim metodama su vrlo bliski pojmovi ,iako se koriste u dvije različite programske paradigme. Naime biblioteka je skup funkcionalnosti koje pozivamo. Izrađuju se kada dijelove koda konstantno ponavljamo u više različitih programa. Kako ih ne bismo svaki puta prepisivali i time činili naš kod nečitkim, stvaramo biblioteku koja će ,kada mi to zatražimo pozvati određenu funkciju i nama vratiti željeni rezultat. [1][3]

Biblioteke su upravo math i cmath u Phytonu i cmath u C++.

d. Primjer biblioteke

Pojma biblioteke možda će se bolje shvatiti iz primjera:

```
#Phyton kod
konstanta = 3.1415
def zbroji(a,b):
    return a+b


def oduzmi(a,b):
    return a-b


def na_potenciju(a,b):
    return pow(a,b)
```

4. Klasa Math u programskom jeziku Java

U ovom ćemo se odlomku detaljnije upoznati sa matematičkom klasom programskog jezika Java. U prvom dijelu je navedena sažeta dokumentacija same klase. Nakon dokumentacije slijedi primjer primjene u računalnim algebarskim sustavima.

a. Opis klase Math

Klasa Math sadrži statičke metode koje izvršavaju neke osnovne algebarske operacije. Unutar klase postoje metode za izračun vrijednosti eksponencijalnih, logaritamskih i trigonometrijskih funkcija za određeni argument. Klasa Math koristi samo primitivne tipove podataka.

Postoji klasa StrictMath koja vraća dosljednije rezultate. U klasi Math mnoge metode jednostavno delegiraju posao koji trebaju obaviti upravo identičnim metodama u StrictMath razredu. Zbog velike sličnosti klasa Math i StrictMath u ovom odlomku obrađivati će se samo klasa Math.

Za operacije koje izvršava klasa Math koristi tehniku dvojnog komplementa. Važno je naglasiti da je u određenim operacijama moguće prijeći granicu domene brojeva koji određeni primitivni tip koristi. Ako se takav scenarij dogodi neke metode će vratiti ArithmeticException iznimku koja će signalizirati da rezultat nije u domeni primitivnih tipova.

Ovdje ću navesti samo neke od metoda i njihove povratne vrijednosti. Za ostatak upućuje se na cjelovitu dokumentaciju:[4]

Tablica 4.1 Metode, povratne vrijednosti i opisi metoda klase Math

Povratna vrijednost	Statička metoda	Opis metode
double/int/float/long	abs(double/int/float/long a)	Vraća pozitivnu vrijednost argumenta a
double	ceil(double a)	Vraća najmanju cjelobrojnu vrijednost veću ili jednaku argumentu a
double	log1p(double x)	Vraća vrijednost

		prirodnog logaritma
double	pow(double a, double b)	Vraća vrijednost broja a na eksponent b
double	sqrt(double a)	Vraća drugi korijen od argumenta a
double	cos(double a)	Vraća vrijednost kosinusa za argument a(a je u radijanima)
double	atan2(double y, double x)	Vraća vrijednost kuta theta iz konverzije iz pravokutnih koordinata(x,y) u polarne (r,theta)
int/long	addExact(int/long a, int/long b)	Vraća vrijednost zbrajanja a sa b s tim da ukoliko se prijeđe domena primitivnog tipa int/long baca iznimku ArithmeticException

b. Primjer

Za primjer ćemo napraviti implementaciju binomne formule koja glasi:

Jednadžba 4.1 Binomna formula

$$(x+a)^n = \sum_{k=0}^{n} \binom{n}{k} x^k a^{n-k} \tag{4.1}$$

```
import java.lang.Math;


public class PrimjerJava {


	public static double binomnaFormula(double x, double y,
	short exp) {

		double rezultat = 0;

		for(short k = 0; k <= exp; k++) {

			rezultat += nPovrhK(exp, k) * Math.pow(x, k) *
		Math.pow(y,(exp-k));

		}

		return rezultat;

	}


	private static double nPovrhK(short n, short k) {

		double rez = 1;

		for(short i = 0; i < k; i++) {

			rez *= ((double)n-i)/(k-i);

		}

		return rez;

	}
```

Iako ovaj kod ne izgleda komplicirano upravo se na njemu očituje potreba za računalnim algebarskim sustavima i važnostima svih njegovih značajki. Ponajviše nam za ovakve kodove treba upravitelj memorijom. Zašto? Zamislite da za varijablu exp stavite broj 32767 (najveći broj u domeni varijable tipa short).

Pogledajmo sada biblioteke cmath i math iz jezika Phyton.

5. Biblioteke math i cmath programskog jezika Phyton

U narednom odlomku bavit ću se programskim jezikom Phyton te primjenom biblioteka cmath i math. U paragrafu a. biti će objašnjene same biblioteke i njihove namjene te navedene neke od najvažnijih funkcija. Paragraf b. dati će primjer rada sa bibliotekama.

a. Opis biblioteka math i cmath

Biblioteke math i cmath sadrže neke od najvažnijih funkcija i konstanti koje se koriste. Neke od njih su: eksponencijalne ,trigonometrijske

, logaritamske, hiperboličke itd.

Razlika između biblioteka math i cmath kada se radi o funkcijama nije velika. Biblioteka cmath koristi se za rad sa kompleksnim brojevima.

Dodatno sadrži funkcije za pretvorbu kompleksnih brojeva u različite oblike. Povratni tip funkcija biblioteke cmath uvijek je kompleksni broj.

Kako svaki korisnik jezika Phyton nema nužno znanja iz kompleksnih brojeva, radi jednostavnosti, napravljena je biblioteka math koja prima samo realne brojeve.

Ovdje ću navesti samo neke funkcije obaju biblioteka. Kako u Phytonu nije potrebno brinuti se oko povratnih tipova funkcija njih neću navoditi. Za opširnije upućujem na literaturu na u zadnjem odlomku.[6][7]

Tablica 5.1 Funkcije i opisi funkcija biblioteke cmath

Funkcija	Opis funkcije
polar(x)	Vraća polarni oblik kompleksnog broja x
rect(r,phi)	Vraća pravokutni oblik kompleksnog broja zadanog sa polarnim koordinatama r i phi
exp(x)	Vraća vrijednost eksponencijalne funkcije e^x
sin(x)	Vraća vrijednost sinusa u x
log(x[,base])	Vraća vrijednost logaritma po bazi base od x
isinf(x)	Vraća vrijednost true ako je bilo koji dio kompleksnog broja x beskonačan
cosh(x)	Vraća vrijednost hiperboličkog kosinusa za x

Tablica 5.2 Funkcije i opis funkcija biblioteke math

Funkcije	Opis Funkcija
isfinite(x)	Vraća true ako i samo ako je broj x konačan
acos(x)	Vraća vrijednost funkcije arkus kosinus broja x
gamma(x)	Vraća vrijednost gama funkcije u x
radians(x)	Uz pretpostavku da je x u stupnjevima vraća vrijednost kuta u radijanima

b. Primjer korištenja cmath biblioteke

Primjer koji potvrđuje jednakost

Jednadžba 5. 1 Eksponencijalni zapis kompleksnog broja

$$|r|e^{i\varphi} = |r|[\cos(\varphi) + isin(\varphi)] \quad (5.1)$$

```
import cmath


def eulerovaFormula (r,phi):
    c = r*cmath.exp(phi*1j)
    return c


r = 5;
phi = 0.95;


rezultat = eulerovaFormula(r,phi);


print(cmath.polar(rezultat));
```

Ispis:

(5.0, 0.95)

c. Primjer korištenja biblioteke math

Do sada smo samo koristili funkcije iz biblioteka odnosno klasa, no kako su te funkcije implementirane? Računalo sigurno ne može samo izračunati npr. e^x . Ovdje je dan primjer kako je e^x implementiran unutar same biblioteke math. Na kraju dobiveni rezultat ćemo usporediti sa konstantom
koja je dio biblioteke.

Jednadžba 5.2 Taylorov zbroj: aproksimacija funkcije e^x

$$e^x = 1 + \frac{x}{1!} + \frac{x^2}{2!} + \frac{x^3}{3!} + \cdots, \qquad -\infty < x < \infty \tag{5.2}$$

```
import math

broj_clanova = 150

def taylorovZbrojE(x):
    rezultat = 0
    for i in range(0,broj_clanova+1):
        rezultat += math.pow(x,i)/faktorijel(i)
    return rezultat

def faktorijel(n):
    rezultat = 1
    if n == 0:
        return  1
    for i in range(1,n+1):
        rezultat *= i
    return rezultat

e = taylorovZbrojE(1)
print(e)
print(math.exp(1))
```

Ispis :

2.7182818284590455

2.718281828459045

Važno je dodati da ,kako se e^x može preko Taylorovog reda unijeti u računalo ,tako to mogu i ostale funkcije npr. $\cos(x)$, $\sin(x)$ itd.

Slijedi nam obraditi još programski jezik C++ i njegovu biblioteku cmath

6. Biblioteka cmath programskog jezika C++

U ovom odlomku proanalizirat ću korištenje jezika C++ i njegove matematičke biblioteke cmath. U paragrafu a. biti će objašnjeno korištenje biblioteke cmath te navedene i objašnjene neke od funkcija u biblioteci. U paragrafu b. primijenit ćemo neke funkcije iz biblioteke cmath na jednom primjeru.

a. Opis biblioteke cmath

Biblioteka cmath naslijeđena je iz programskog jezika C. U njoj su sadržane neke od najvažnijih funkcija u matematici.

Ovdje se kao i u Phytonovim bibliotekama odnosno Javinoj klasi mogu pronaći trigonometrijske, logaritamske, eksponencijalne i hiperboličke funkcije kao i ostale funkcije koje nam mogu reći nešto o samom argumentu. Mnoge implementacije funkcija u Javinoj klasi i Phytonovim bibliotekama uzete su upravo od biblioteke cmath odnosno njoj preteče biblioteke math.h programskog jezika C .

Važno je naglasiti da ovdje svaka funkcija prima isključivo argumente tipa double te vraća argument istog tog tipa. Navodim samo neke funkcije.[5]

Tablica 6.1 Funkcije i opis funkcija biblioteke cmath

Funkcija	Opis funkcije
cos(double x)	Vraća vrijednost funkcije kosinus za broj x
sinh(double x)	Vraća vrijednost hiperboličke sinus funkcije za broj x
pow(double x, double y)	Vraća vrijednost broja x na potenciji y
log(double x)	Vraća vrijednost prirodnog logaritma za broj x

b. Primjer korištenja biblioteke cmath

U ovom ćemo se primjeru upoznati sa integriranjem u računalu. Važno je naglasiti da je računalo isključivo digitalan sustav. Zbog te činjenice nije moguće u računalo unijeti sve brojeve u nekom realnom intervalu [a,b] jer tih brojeva ima beskonačno mnogo. Kako onda integrirati neku funkciju?

Funkcije se u računalu ne mogu integrirati ,ali kako znamo da integral predstavlja površinu ispod krivulje možemo uzeti zbroj jako puno uzoraka. Integrirat ću u ovom primjeru funkciju kosinus na intervalu [-π ,π].

Jednadžba 6.1 Integral funkcije kosinus u granicama [-π, π]

$$\int_{-\pi}^{\pi} \cos(x)dx \tag{6.1}$$

```
#include<iostream>

#include<cmath>


#define UZORAK 0.00001

#define PI 3.1415926


using namespace std;


double integral(double low, double high,
double(*funkcija)(double)) {

      double rezultat = 0;

      for (double i = low; i <= high; i = i + UZORAK) {
```

```
            rezultat += (*funkcija)(i);
      }
      return rezultat * UZORAK ;
}


int main() {

      double high = PI;
      double low = -PI;

      double rezultat = integral(low, high, cos);
      cout << rezultat << endl;
      system("pause");
      return 0;
```

7. Zaključak

Računalni algebarski sustavi, kako je kroz čitav seminar pokazano, vrlo su složeni. U navedenim primjerima naveden je tek mali broj algoritama koje moraju sadržavati biblioteke algoritama. Biblioteke CAS-a također sadrže i funkcije iz biblioteka ili klasa s kojima smo se upoznali([4][5][6][7]).

Važno je naglasiti kako sa velikim brojem međurezultata moramo moći upravljati. Upravo zbog toga u računalnom algebarskom sustavu moramo imati dobar upravitelj memorijom. U to smo se uvjerili u primjeru integriranja (Jednadžba 6.1), Taylorovog zbroja (Jednadžba 5.2) te u binomnoj formuli (Jednadžba 4.1). Naravno, bez samih programskih jezika te njihovih prevoditelja ne bismo mogli ostvariti CAS.

Korisničko sučelje računalnih algebarskih sustava važna je stavka jer upravo preko njega mi kao korisnici sustava unosimo željene formule te dobivamo rezultat.

Naglasio bih još da uz pomoć svih navedenih stavki računalnog algebarskog sustava danas možemo u vrlo kratkom vremenu dobiti rješenja nekih ne tako jednostavnih matematičkih problema. Upravo su CAS-ovi pomogli u mnogim važnim otkrićima suvremenog doba.

8. Literatura

[1] **THE C++ PROGRAMMING LANGUAGE – BJARNE STOURSTRUP**

[2] **PROGRAMIRANJE U JAVI – MARKO ČUPIĆ**

[3] http://thepythonguru.com/

[4] https://docs.oracle.com/javase/8/docs/api/java/lang/Math.html

[5] http://www.cplusplus.com/reference/cmath/

[6] https://docs.python.org/3/library/math.html

[7] https://docs.python.org/3/library/cmath.html#module-cmath

9. Sažetak

Seminarski rad opisuje algebru i njezinu primjenu u sklopu računalnih algebarskih sustava. Kako se svaki računalni algebarskih sustav sastoji između ostalog i od programskog jezika i njegovog prevoditelja, u seminaru je opisana algebra unutar programskog jezika.(1)

Zbog velikog broja programskih jezika, kao reprezentante, odlučio sam odabrati jezike Phyton, Javu i C++. Ta tri programska jezika odabrao sam jer imaju najveću primjenu i najviše korisnika. Kroz njih je detaljnije opisana primjena algebre u programskim jezicima.

U nastavku seminara opisao sam osnovne operacije unutar programskih jezika kao što su zbrajanje, oduzimanje, množenje, dijeljenje i ostatak pri cjelobrojnom dijeljenju.(2)

Potom sam pokušao što kraće objasniti pojmove biblioteka i klasa i kako ih koristiti. (3)

Nakon objašnjenja što su to uopće biblioteke i klase opisao sam korištenje matematičkih biblioteka , odnosno razreda unutar navedenih programskih jezika.(4,5,6)

Kroz seminar sam koristio nekoliko matematičkih jednadžbi kao primjer korištenja algebre u programskim jezicima .(Jednadžba 2.1,Jednadžba 4.1,Jednadžba 5.1,Jednadžba 5.2,Jednadžba 6.1). Također sam ,kako ne bih pisao i opisivao svaku funkciju iz pojedinih biblioteka i klasa , u literaturi naveo poveznice na službene dokumentacije klasa.([4][5][6][7])