Izvještaj

U ovom izvještaju osvrnut ću se na detalje implementacije klase „Polinom“ u sklopu Projekta 1 iz predmeta Strukture podataka i algoritmi. Jedan od nedostataka korištenja niza kao strukture podataka kojom se implementira polinom je preveliko rasipanje memorije. Implementacija polinoma preko niza podrazumijeva da za svaki eksponent manji ili jednak stepenu polinoma čuvamo koeficijent u određenom mjestu u nizu. Pa čak i ako je taj koeficijent jednak nuli. Takav način troši prilično mnogo memorije u slučaju kada polinomi imaju samo nekoliko članova (monoma) čiji se eksponenti mnogo razlikuju. Također, algoritmi koji rade sa takvim polinomima su sporiji, jer se većina posla svodi na nepotrebne operacije sa nulama. Zbog toga se nameće ideja da u memoriji čuvamo samo one članove polinoma koji zaista postoje, tj. čiji su eksponenti različiti od nule. No, kako mi ne znamo unaprijed ništa o mogućim članovima polinoma, mi ih moramo dinamički dodavati. Dakle, najpametniji izbor strukture podataka je zapravo povezana lista. Čvor u listi će nam predstavljati jedan član polinoma. Dakle, jedan čvor treba da u sebi sadrži informaciju o eksponentu i ta informacija je cijeli broj, te informaciju o koeficijentu, tj. jedan realan broj. Osim toga, čvor treba da sadrži i vezu. Za potrebe ove klase „Polinom“, ispotavlja se da je korisno koristiti dvosturko povezanu listu, što implicira da struktura Čvor treba da sadrži dvije veze, na prethodni i sljedeći element. Ja sam odlučio čuvati Čvorove, tj. članove polinoma, u sortiranom opadajućem poretku prema eksponentu. Takav pristup olakšava i ubrzava neke metode, dok komplikuje neke druge. Dakle, sama klasa „Polinom“ zapravo predstavlja samo pokazivač na prvi čvor. Ja sam osim toga uveo i jedan dodatni atribut koji se poslije pokazao korisnim, broj čvorova u polinomu. Dakle, moja klasa „Polinom“ ima dva atributa, pokazivač na prvi čvor (dakle čvor sa najvećim eksponentom) i broj čvorova u polinomu. Klasa koju sam ja implementirao ima nekoliko konstruktora.

Prvi konstruktor je bez parametara i on zapravo stvara „nul-polinom“. Tehnički, on samo postavlja vrijednost pokazivača „početak“ na nulu, te broj članova u polinomu također postavlja na nulu.

Dalje, implementirao sam i konstruktor sa jednim parametrom, realnim brojem. Taj konstruktor pravi novi polinom sa samo jednim članom i to slobodnim članom polinoma. Njega sam realizirao uz pomoć metode „dodajČlan“ koju ću sada objasniti.

Metoda „dodajČlan“ je jedna od najvažnijih i najzahtjevnijih metoda u cijeloj klasi. Ona ima dva parametra, realan broj k i cijeli nenegativan broj e. Njen zadatak je da u polinom doda član kx^e. Ukoliko je k=0, zapravo ne dodajemo član, pa taj slučaj možemo zanemariti, dakle ne treba vršiti nikakve izmjene na polinomu. Osim toga, razlikovao sam nekoliko slučajeva. Prvi slučaj je kada je polinom prazan i mi ubacujemo novi član u njega. Tada je potrebno samo stvoriti novi čvor sa zadanim parametrima, postaviti da obje njegove veze budu nul-

pokazivači i pokazivač „početak“ postaviti da pokazuje na novostvoreni čvor. Dakle, sada imamo jedan čvor u listi. Sljedeći slučaj je kada ubacujemo član čiji je eksponent veći od do sada najvećeg eksponenta. To je izolovan slučaj jer ako dođe do njega, moramo mijenjati vrijednost pokazivača „početak“, osim uobičajenih promjena pokazivača. Početak liste će nam u ovom slučaju biti novoformirani čvor. I treći slučaj je kada lista nije prazna i kada se ubacuje čvor koji nema najveći eksponent. Tada je potrebno kretati se kroz listu sve dok ne naiđemo na neki čvor sa manjim eksponentom od onog kojeg ubacujemo. Kada naiđemo na takav čvor, znači da smo pronašli mjesto za novi čvor. No, to ne znači nužno da moramo praviti novi čvor. Moguće je da je već postojao čvor sa eksponentom e i da samo treba promijeniti njegov koeficijent. Ukoliko je to slučaj, ne trebamo stvarati novi čvor, u suprotnom moramo stvoriti novi čvor i promijeniti odgovarajuče pokazivače. Bitno je napomenuti jednu stvar u slučaju da je već postojao čvor sa eksponentom e. Moguće je da nakon promjene njegovog koeficijenta, taj koeficijent bude 0, tj. da dođemo u sitaciju da nam taj čvor više nije potreban. U tom slučaju ga je potrebno obrisati, što radimo pomoću metodi „obrisiClan“ koju ćemo objasniti kasnije. Time su pokriveni svi slučajevi kod dodavanja novog člana. Naravno, prilikom umetanja novog člana, potrebno je ažurirati i odgovarajući atribut o broju članova polinoma. Primjetimo da je ova metoda „pametna“, tj. sama se pobrine da polinom ostane do kraja sređen, sortiran prema eksponentima i bez nepotrebnih čvorova. Nju možemo koristiti kasnije, znajući njene osobine.

Također, za klasu „Polinom“ je neophodno bilo implementirati i konstruktor kopije. On je prilično trivijalan. Potrebno je samo kretati se kroz polinom koji nam je parametar konstruktora kopije i dodavati nove čvorove na kraj polinoma koji kreiramo. To se lagano uradi u jednoj petlji koja kreće od početka polinoma p (parametra) i koja se vrti sve dok je tekuci pokazivač različit od nule. U tijelu petlje ćemo stvoriti novi čvor po uzoru na onaj na koji nam pokazuje pokazivač „tekući“ i dodati taj novi čvor na kraj polinoma koji stvaramo.

Preklopljeni operator dodjele je gotovo identičan konstruktoru kopije uz jednu malu izmjenu. Prvo je bilo potrebo vodit računa o „samododjeli“, dakle ukoliko stavimo nešto tipa „a=a“, da se ne uništi objekat „a“. Zato operator dodjele ne treba ništa da radi u tom slučaju, jer a već jeste jednako a. U suprotnom, prvo treba obrisati sve čvorove koji se trenutno nalaze u polinomu, pa onda staviti nove čvorove, po uzoru na parametar operatora dodjele. Za brisanje nam služi funkcija „obrisiClan“ koju pozivamo nad pocetkom sve dok mozemo. Kada dodjemo u situaciju da je pocetak jednak nul-pokazivaču, to znači da smo izbrisali sve čvorove i onda samo obavimo isti posao kao i u konstruktoru kopije.

Destruktor je također jako bitan a normalno funckionisanje klase. U njemu jednostavno brišemo čvorove s početka sve dok je to moguće. U petlji u kojoj vršimo brisanje, moramo imati dva pokazivača, jedan na čvor koji ćemo obrisati, ali i jedan na njegovog sljedbenika, jer moramo nekako imati pristup listi i kad obrišemo njen početak. Brisanje se vrši sve dok ima

čvorova u listi. Kada obrišemo sve čvorove zapravo smo oslobodili svu memoriju koju je polinom dinamički alocirao i tako je destruktor završio svoj posao.

Metoda koja briše član polinoma zove se „obrisiClan“ i ima kao parametar pokazivač na čvor koji brišemo. Potrebno je razlikovati nekoliko slučajeva. Prvi slučaj je da je čvor koji brišemo početak. Ukoliko jeste, imamo dva podslučaja. Ukoliko brišemo početak, a on je ujedno i jedini element liste, potrebno je samo obrisati početak i staviti pokazivač „pocetak“ na nulu, i ažurirati atribute tako da dobijemo nul-polinom. Ukoliko brišemo početak, a ima još članova osim njega, potrebno je ažurirati pokazivač „početak“ tako da pokazuje na sljedbenika dosadašnjeg početka. Nakon promjene odgovarajućih pokazivača obrisat ćemo početak. I treći slučaj je da ne brišemo početak. U tom slučaju je potrebno ažurirati informacije o vezi prethodnika i sljebenika čvora koji brišemo. Nakon toga jednostavno obrišemo čvor. Nakon svega toga je potrebno ažurirati atribur „brojCvorova“ tako što ćemo ga smanjiti za 1.

Binarni operator sabiranja je jednostavan. Algoritam je sljedeći: stvoriti novi polinom po uzoru na prvi sabirak. To se može uraditi konstruktorom kopije. Taj novi polinom će nam biti rezultat na kraju algoritma. U taj rezultat je potrebno samo dodati sve članove iz drugog sabirka. Obzirom da je metoda „dodajClan“ pametna, jednostavno prođemo petljom kroz sve članove drugog polinoma i prosto ih metodom nadodamo na rezultat. Zbog učinka metode „dodajClan“, krajnji rezultat će biti do kraja sređen, te ćemo imati novi polinom u sortiranom opadajućem poretku prema eksponentima.

Binarni operator oduzimanja je analogan onom za sabiranje, samo što članove iz drugog polinoma dodajemo u prvi sa izmijenjenim predznakom koeficijenata.

Binarni operator množenja je nešto komplikovaniji. Treba primjetiti da je metoda „dodajClan“ u najgorem slučaju linearna po broju članova polinoma. Dakle, ako dodajemo najmanji član u polinom ta metoda će proći kroz cijeli polinom. Ukoliko dodajemo najveći član, onda ta metoda ima konstantno vrijeme izvršavanja. Za potrebe množenja ćemo prvo stvoriti novi prazan polinom koji će nam služiti da u njega stavljamo rezultat množenja. Pošto se polinomi množe tako što se izmnože svi članovi prvog sa svim članovima drugog polinoma, takav je i algoritam u ovoj klasi. Jednom petljom prolazimo kroz sve članove prvog polinoma, dok unutar nje imamo jednu petlju za sve članove drugog polinoma. Kada imamo dva fiksirana člana, da bi ih pomnožili, trebamo pomnožiti njihove koeficijente i sabrati eksponente. Takav član dodamo na rezultat. U cilju nam je da dodajemo članove sa što većim eksponentima, zato petlje ide od početka prema kraju polinoma, jer su polinomi upravo sortirani u opadajućem poretku prema eksponentima.

Binarni operatori dijeljenja i ostatka pri dijeljenju su gotovo identični. Zapravo u oba slučaja moramo podijeliti dva polinoma, a pri dijeljenju je potrebno čuvati i količnik i ostatak. Kod binarnog operatora dijeljenja, kao rezultat ćemo vratiti količnik, kod binarnog operatora

ostatka pri dijeljenju ćemo vratiti ostatak kao rezultat. Uveli smo dva polinoma q i r koji će redom u svakom trenutku predstavljati trenutni količnik i trenutni ostatak pri dijeljenju. q postavljamo na nulu jer, dok r postavljamo da bude polinom koji dijelimo. Algoritam je samo primjena postupka dijeljenja polinoma. Uzmemo prvi član polinoma r i podijelimo ga sa prvim članom polinoma kojim dijelimo. Taj količnik nadodamo na rezultat dijeljenja, tj. polinom q. Dalje, pomnožimo sa tim količnikom cijeli polinom p (kojim dijelimo) i takav prozivod oduzmemo od polinoma r. Tako će se sigurno smanjiti stepen polinoma r. Postupak ponavljamo sve dok ne dobijemo da nam je r nul-polinom ili polinom stepenja manjeg od polinoma p (polinoma kojim dijelimo). Jer upravo takav je postupak dijeljenja polinoma. Na kraju algoritma, q nam je količnik dok nam je r ostatak. Treba naglasiti, da ako dijelimo sa polinomom koji je većeg stepena od dijeljenika, onda je rezultat 0, a ostatak pri dijeljenju je sam dijeljenik. To je jedan poseban slučaj. Također, poseban slučaj je i kad dijelimo sa nul-polinomom, jer dijeljenje sa nulom nije definisano, pa moj operator u tom slučaju baca izuzetak.

Operatori “+=“, “-=“, “*=“, “/=“, “%=“ su trivijalni budući da se zasnivaju na već implementiranim operatorima “+“, “-“, “*“, “/“, “%“.

Unarni operator „+“ je također trivijalan budući da on ne radi ništa, potrebno je samo vratiti kao rezultat polinom nad kojim je i pozvan.

Unarni operator „-“ je nešto složeniji. Kod njega pravimo novi polinom koji je na početku prazan. Onda krenemo od početka polinoma nad kojim se poziva operator i svaki član samo nadodamo na rezultat ali sa izmijenjenim predznakom. Ne moramo koristiti metodu „dodajClan“, tj. njeno korištenje bi bilo pogubno, obzirom da uvijek dodajemo najmanji element i uvijek bi imalo worse-case scenario. Umjesto toga svaki član prosto dodajemo na kraj liste, tako što ćemo dinamički stvoriti novi član i promijeniti odgovarajuće pokazivače. Također je potrebno pamtiti gdje se trenutno nalazi kraj liste kako bi uvijek bili u mogućnosti dodati element na kraj. Kada završimo sa time, jednostavno vratimo kao rezultat iz funkcije novoformirani polinom. On će biti isti kao i onaj nad kojim je pozvan operator, ali sa izmijenjenim predznacima.

Operatori poređenja su prilično jednostavni. Prvo, primjetimo da su nam dostupni brojevi clanova polinoma. Ukoliko dva polinoma imaju razlicit broj clanova, oni ne mogu biti jednaki. Ukoliko im je broj članova isti krećemo petljom od početka oba polinoma i provjerajemo da li su članovi na odgovarajućim pozicijama jednaki. Ukoliko nisu, to znači da su polinomi različiti i odmah možemo prekinuti sa testiranjem jednakosti. Ako prođemo kroz cijeli polinom bez da smo naišli na različite članove, onda su polinomi jednaki. Analogan je postupak za provjeru različitosti dva polinoma.

Moj operator za ispis se oslanja na funkciju „ispisiClan“. Ta funkcija odvojeno tretira prvi član polinoma i sve ostale. Zato imam poseban parametar „prvi“ koji mi govori da li je neki član polinoma prvi ili nije. Funkciju „ispisiClan“ pozivam sa true vrijednosti parametra „prvi“ samo za početak liste, u ostalim slučajevima sa false. U toj funkciji treba razmatrati različite vrijednosti za eksponent i koeficijent. Ukoliko je neki član prvi, eventualni plus se ne treba ispisati. Ukoliko je član pozitivan a nije prvi, plus se treba ispisati. Ukoliko je eksponent jednak jedinici, on se ne mora pisati (prirodnije je „6x“ umjesto „6x^1“). Također, ukoliko je eksponent jednak nuli, uopšte ne moramo pisati „x“, jer to je slobodni član („5“ je pravilno, dok „5x^0“ i nije baš). Također, ako je koeficijent jednak 1, ne moramo ga pisati (normalnije je „x^2“ umjesto „1x^2“). U funkciji sam tretirao sve ove specijalne slučajeve. Onda sam u operatoru za ispis samo prošao kroz sve članove polinoma i pozvao funkciju „ispisiClan“ nad njima.

Operator za unos polinoma je prilično jednostavan. Koristeći metodu „peek()“ iz klase istream lagano provjerajemo da li smo došli do kraja unosa. Sve dok ne dođemo do kraja unosa, unesemo realan broj, pa dva znaka, tj. znakove „x“ i „^“ pa nakon toga cijeli broj. Takav je format unosa polinoma zadan u tekstu projetka. Nakon toga samo iskoristimo metodu „dodajClan“ da uneseni član i dodamo u polinom.

Metode „vratiStepen“, „vratiBrojClanova“ i „daLiJeNula“ su trivijalni getteri i njih ne treba posebno objašnjavati.