Tema 1.: RealanBroj

Vedad Delić, TKN

Projekat 1

Strukture podataka i algoritmi

Naš zadatak u ovom projektu je da implementiramo klasu RealanBroj koja podržava operacije sa realnim brojevima proizvoljne veličine. Klasu implementirati kao dvije liste, od kojih jedna predstavlja cifre prije decimalnog zareza, a druga cifre nakon zareza. Naravno elementi obje liste su tipa int.

U public dijelu klase nalaze se definicije operatora koje je potrebno preklopiti. Potrebno je preklopiti unarni operator –, kao i binarne operatore +, −, ∗, +=, −=, ∗=, te ostream i instream operatori << i >>. Neke funkcija koje su incijalizirani kao prijateljice ove klase su:

```
friend Realni operator+ (Realni& lhs, Realni& rhs);

friend Realni operator- (Realni& lhs, Realni& rhs);

friend ostream& operator<<(ostream& out, Realni& Rezultat);
```

a to je učinjeno jer je njima neophodan sadržaj privatnog dijela klase. Također je radi jednostavnosi u ovom dijelu klase odmah iplementiran i unarni op. -.

U private dijelu klase nalaze se :

```
Lista<int> Prije;

Lista<int> Poslije;

bool je_li_pozitivan;

Realni(){}
```

Znači primjetimo dvije liste koje čuvaju cifre prije i poslije dec. zareza (liste su uključene iz posebnih file-ova koji su dostavljeni). Također klasa RealniBroj je nazvana radi jednostavnosti Realni. Dostavljam i file- ove: main (koji nije neophodan ali u slučaju pada evaluatora moguće je dostalakše provjeriti ispravnost pojednih funkcija), lista.h, lista.cpp, Realni.cpp (zbog nekog razloga compiler ne može dokučiti ovaj file, iako je u istom projectu sve, istom folderu,... imajući u vidu da i nije najbolja praksa ovaj dio sam komentarisao i sve radio u .h (header) file- u, gdje sve radi ok. Nakon dvoumljena da li ovaj file uopće poslati odlučio sam se za to iako predstavlja puku kopiju.), Realni.h, te evaluator.cpp (adaptirani kod, koda sa vježbi koji je u suštini zadužen za izvođenje samog računa izraza. Zbog obimnosti programa i veće mogućnosti za grešku dostavio sam i main.cpp file za lakše izvođenje testiranja).

Prvo opišimo šta je rađeno u private dijelu:

- Lista<int> Prije;
    - Obična lista tipa int koja je zadužena da čuva cifre prije dec. zareza i to jednu po jednu.
- Lista<int> Poslije;
    - Obična lista tipa int koja je zadužena da čuva cifre poslije dec. zareza i to jednu po jednu.
- bool je_li_pozitivan;
    - Logička varijabla tipa bool čiji je cilj da nam vrati log. vr. True ukoliko je uneseni decimalni broj veći ili jednak 0, inače vraća log. vr. False. Ova varijabla je uvedena radi izvođenja kasnijih operacija o čemu će kasnije biti govora.
- Realni(){}
    - Konstruktor bez parametara, smješten u private dijelu jer nije zahtijevan zadatkom ali je neophodan za izvođenje nekih operacija na način za koji sam se odlučio.
- Napomenimo da nisam iplementirao destruktor jer pozivam liste za koje destruktor već postoji tako da ne bih trebao imati problem, bar što se tog dijela tiče.

Sada opišimo sta je rađeno u public dijelu

- Realni(double broj);
    - Konsturktor sa jednim parametrom koji se traži postavkom. Metoda koja je korištena kod implementacije ovog konstruktora potiče još iz prog. 1. Razdvajam broj na int dio i na decimalni dio. Opis za cjelobrojni dio: Uzimam poslj. Cifru i dodajem je na kraj do tad prazne liste. Broj od kog sam uzeo poslj. Cifru skraćujem za jednu cifru. Lista više nije prazna. Postupak ponavljam sve dok zadani broj ne postane manji od 1. Sada pređimo na dec dio: Uzimam samo decimalni dio na slj način:

```
int c=broj;

double trenutni=broj-c;

if (trenutni==0) return;
```

      Sad množim sa 10 i dodajem na početak liste cifara Poslije. Ako trenuti postane =0 to znači da su svi brojevi nakon 0 i tu se zaustavljamo.

- Realni::Realni(string broj)
    - Konsturktor sa jednim parametrom koji se traži postavkom. Metoda koja je korištena kod implementacije ovog konstruktora potiče još iz prog. 1. Glavni cilj je provjera da li je broj cifra, što i nije teško uraditi i da li postoji više od jedan zarez. Ukoliko prvi usl nije zadovoljen a drugi jeste bacam izuzetak koji nijepoželjno hvatati u konstruktoru pa ga ostavljam korisniku da ga može uhvatiti kad mu to zatreba. Ostatak koda je logički sličan konstruktoru koju prima double. Napomena: korišten sljedeći trik npr '9'(char) ukoliko oduzmemo char npr '0' dobijamo int. '9' - '0' = int 9.

- ostream& operator<<(ostream& out, Realni& Rezultat)
    - operator ispisa(izlaznog toka) čija je iplementacija prilično očigledna i ne zahtijeva previše detaljan opis. Vodilo se računa da li je broj negativan i da zarez bude na ispravnom mjestu.
- ostream& operator<<(istream& in, Realni& upisi)
    - operator upisa čija iplementacija na izgled i nije trivijalna. Vodilo računa se da se ispravno uzme -, i . . ako nesto nije ispravno uneseno (npr slovo) da odmah stane sa gledanjem ostatka. Kada je prazno mjesto uneseno ili enter da uzme

iz toka sta je uneseno i vrsi dalje operacije preklapanje sa tim. Nule se brišu u konstruktoru, pa taj dio nije ponovljen. Vodilo se strikto računa da li je broj uneseni, tj. U ovom slučaju char između '0' i '9'(uz vec navedeno – i .)na kraju je povratni tip in(input / ulaz).

- Realni& operator- ()
    - Unarni operator oduzimanja čija je iplementacija direktno data u public dijelu klase zbog njegove jednostavnosti. Njegov cilj je da promjeni predznak broja nad kojim je pozvan, npr. Primjenimo li ga nad -5.31 dobijamo vraćeno 5.31

- friend Realni operator+ (Realni& lhs, Realni& rhs);
    - Binarni operator sabiranja. Prvo sam brojao decimale jednog pa grugog broja. Onaj broj koji ima manjecifri sam nadopunio nulama dok ga nisam izjednačio sa ciframa onog drugog. Potom pravim bool varijablu ima_li_prenosa koja ukoliko je zbir cifri na odgovarajucim mjesima >9 ima vrijednost True. Potom u narednoj liniji sabiram cifre na odgovarajućim mjestima ali i bool ima_li_prenosa koja zbog automatske konverzije ima u ovom slučaju 1 ili 0. Također liste su pravljenje 'naopako', unazad radi jednostavnosti i jasnije logičke izrade.

- Realni operator- (Realni& lhs, Realni& rhs)
    - Binarni operator sabiranja. Prvo sam brojao decimale jednog pa grugog broja. Onaj broj koji ima manjecifri sam nadopunio nulama dok ga nisam izjednačio sa ciframa onog drugog. U suštini ovaj dio sam podijelio na nekoliko dijelova što ga razlikuje od sabiranja. To ću najbolje objasniti na primjeru, gdje su Realni a, b; -a-b=-(a+b); a+(-b)=a-b; -a+b= b-a; -a-(-b)=b-a; za a>b .. a-b; a-(-b)=a+b; -a-b=-a+b; b(-a) –a+b;
        - U ovom dijelu će nam dosta olakšati i funkcija koja provjerava za dva Realni broja koji je veci po apsolutnoj vrijednosti.

- Realni& operator+= (const Realni& rhs);
- Realni& operator-= (const Realni& rhs);
- Realni& operator*= (const Realni& rhs);
    - Opisujem zajedno (u paketu) zbog dosta slične iplementacije. Ona je prilično trivijalna i podrazumijeva vraćanje 'dodanog' pokazivača, tj u zavisnosti od prirode operatora tj da li će u pitanju biti množenje, sabiranje ili oduzimanje. Ovo je već viđeno u predmetu prog ii te se kao takvo očekuje već usvojenim.
- friend Realni operator* (Realni& lhs, Realni& rhs);
    - U pokušaju iplementacije i preklapanja ovog operatora je korišten poznati karasuba algoritam. Prvo sam napravio nekoliko pomoćnih metoda, budući da alg. Radi sa brojevima predstavljenim kao string i to u binarnom zapisu.

- Ubinarni pretvara broj u binarni
- Dodaj stringove po bitima sabira dva stringa u bin zapisu
- Long int mnozi mnozi dva stringa
- Int mnozi_bazni osvrće se samo na bazicni slucaj mnozenja dva bin broja
- Int izjednaci vodi racuna da dva broja imaju isti broj cifara

Inače, vodio sam se time da sam broj iz dvije liste stavio u jednu cjelinu. Zapamtio broj decimala. Isto uradio i sa drugim brojem. Broju dec od ranije dodao broj novih dec mjesta. Koristeci se vec navedenim metodama vrsio pretvaranja iz liste u string, binarni, ... Na kraju tek pozvao nesto modifikovani i adaptirani algoritam mnozenja stringova u bin zapisu, koji za cilj ima da vrati tip int. Nakon svega tog kreiram novu listu Rezultat i na osnovu vec ranije znanog broja cifara dec mjesta, u listu cifara rezultata prije zareza dodajem sve decimalne cifre, a isto i nakon zareza potom. Te na kraju vraca rezultat klase Realni. Nadam se da logika fje zadovoljava kriterije jer je u nju ulozeno dosta truda i vremena.

- friend Realni operator^ (Realni& lhs, int a);
    - operator je preklopljen tako sto stepenuje Realni broj sa nekim brojem tipa int. Također je i ovdje korištena modifikacija jednog od dobro poznatih algoritama cija je vrem slozenost (log n). Inace sama iplementacija ovog preklapanja je dosta jednostavna jer je koristena funkcija stepenuj koja radi sav posao. Unutar te funkcije se poziva operator * koji ukoliko ne bude radio moze i ovdje da stvori problem. Također se vodilo dosta računa da sa logičkog aspekta bude sto je vise moguce jasnije i jednostavnije rijeseno. Jedan od problema koji se ovdje pojavio je dijeljenje dva realna broja. To sam pokusao izbjeci tako sto sam prvi pomnozio sa drugim na -1.