OBJEKTNO ORIJENTISANO PROGRAMIRANJE

- domaći zadatak broj 1 -

Funkcionalna specifikacija

Na programskom jeziku C++ implementirati statičku biblioteku (.lib) klasa velikih decimalnih brojeva (*BigDecimal*). Potom napisati glavni program (kao konzolnu .exe aplikaciju) koji testira rad sa celim brojevima. U nastavku teksta termin *veliki broj* značava *veliki decimalni broj*.

Specifikacija klase BigDecimal

Klasa `BigDecimal` predstavlja apstraktni tip podataka za rad sa označenim velikim brojevima. Veliki celi brojevi imaju neograničen broj cifara za celobrojni i razlomljeni deo. Sve operacije se obavljaju u neograničenoj tačnosti. Za smeštanje cifara NIJE DOZVOLJENO koristiti kolekcije iz standardne biblioteke, već isključivo ugrađene i sopstevene tipove podataka i strukture. NIJE DOZVOLJENO koristiti biblioteke koje implementiraju slične ili iste funkcionalnosti!

Veliki broj se pravi na osnovu znakovnog niza cifara.

Implementirati konstruktor `BigDecimal::BigDecimal(char* num);`. Argument `num` sastoji se od cifara u celom i razlomljenom delu razdvojenih tačkom. Tačka i razlomljeni deo su opcioni. Na prvom mestu opciono se može pojaviti predznak broja. Broj cifara je neograničen.
Preporučuje se implementacija privatnog uslužnog konstruktora za optimizovano incijalizovanje objekata internim stanjem.

Veliki brojevi moraju biti nepromenljivi (immutable).

Jednom kada se napravi objekat tipa BigDecimal i jednom kad mu se definiše vrednost, NE SME se više menjati. Sve operacije sa velikim brojevima prave nove privremene rezultate/objekte i NE SMEJU menjati vrednosti operanada.

Propisno uništavanje velikih brojeva.

Implementirati propisno uništavanje objekata klase BigDecimal.

Pomeranje decimalne tačke za N mesta.

Veliki broj može da se transformiše pomeranjem tačke za *n* mesta ulevo ili udesno. Rezultat je novi veliki broj sa pomerenom tačkom. Implementirati sledeće uslužne javne funkcije:

```
BigDecimal BigDecimal::shl(int n); // left
BigDecimal BigDecimal::shr(int n); // right
BigDecimal BigDecimal::rmd(int* n); // remove
```

Funkcija **`rmd`** (remove dot) potpuno eliminiše razlomljeni deo i bočnim efektom vraća broj mesta za koliko je tačka pomerena udesno. Logički dobijena vrednost je „ceo" broj.

Sabiranje velikih brojeva.

Implementirati metodu `BigDecimal BigDecimal::add(BigDecimal*);`.
Metoda sabira dva velika broja i vraća novi objekat. Voditi računa o znakovima operanada, kao i znaku rezultata.

Oduzimanje velikih brojeva.

Implementirati metodu `BigDecimal BigDecimal::sub(BigDecimal*);`.
Metoda oduzima dva velika broja i vraća novi objekat. Voditi računa o znakovima operanada, kao i znaku rezultata.

Poređenje velikih brojeva (greater).

Implementirati metodu `bool BigDecimal::greater(BigDecimal* num);`.
Metoda vraća true ukoliko je vrednost broja za koji je pozvana veća od vrednosti broja num.

Poređenje velikih brojeva (less).

Implementirati metodu `bool BigDecimal::less(BigDecimal* num);`.
Metoda vraća true ukoliko je vrednost broja za koji je pozvana manja od vrednosti broja num.

Poređenje velikih brojeva (equals).

Implementirati metodu `bool BigDecimal::equals(BigDecimal* num);`.
Metoda vraća true ukoliko je vrednost broja za koji je pozvana jednaka vrednosti broja num.

Apsolutna vrednost velikog broja (abs).

Implementirati metodu `BigDecimal BigDecimal::abs();`.
Metoda vraća apsolutnu vrednost zadatog broja num kao novi objekat.

Ispisivanje velikih broja u izlazni tok.

Preklopiti operator << za ispisivanje objekta klase `BigDecimal` u izlazni tok, gde je prototip operacije `ostream& operator<<(ostream&, const BigDecimal&)`. Operatorska funkcija ispisuje sve cifre i znakove broja koji postoje.

Test funkcija

U projektu glavnog programa treba da postoji funkcija `void test();` koja testira rad sa velikim brojevima. Studenti treba da implementiraju datu funkciju i uslovno je prevode ako je definisan makro STUDENT_TEST. Predvideti i postojanje makroa PROF_TEST. Ako su oba makroa definisana, PROF_TEST ima prioritet i tada se prevodi tajni test primer koji će biti dat na odbrani. Funkcija main treba samo da pozove test funkciju i na kraju ispiše njeno trajanje u milisekundama, korišćenjem tipova i operacija iz zaglavlja <ctime>.

Tehnički zahtevi i smernice za izradu rešenja

Sve klase i metode moraju biti imenovane prema zahtevima iz domaćeg zadatka. Svaka klasa koja koristi dinamičku memoriju mora biti bezbedna za korišćenje. Programski kod klasa rasporediti u odgovarajuće **.h** i **.cpp** fajlove. Nije dozvoljeno korišćenje globalnih promenljivih za razmenu podataka. Ukoliko u zadatku nešto nije dovoljno jasno definisano, treba usvojiti razumnu pretpostavku i na temeljima te pretpostavke nastaviti izgrađivanje svog rešenja.

VAŽNE NAPOMENE

Za uspešno odbranjen domaći zadatak potrebno je na odbrani pokazati kod podeljen na odgovarajuće projekte, .h i .cpp fajlove.

- Klase kojima su implementirani osnovni koncepti treba da budu smeštene u poseban projekat rešenja koji se prevodi kao statička biblioteka (**`BigDecimal.lib`**).
- Glavni program napisati u posebnom projektu koji se prevodi kao Win32 Console Application (**`testdz1.exe`**) fajl i koji treba povezati sa statičkim bibliotekama. Glavni program treba implementirati tako da pozove globalnu funkciju `void test();`.
- Studenti treba da implementiraju svoju verziju ove funkcije tako da demonstriraju operacije sa velikim brojevima.
- Na kraju programa potrebno je ispisati na standardnom izlazu vreme trajanja funkcije `test` u milisekundama. Može se iskoristiti kod za merenje vremena na jeziku C++ koji je dat u zadatku 2.8 u materijalima za vežbe.
- NIJE DOZVOLJENO SMESTITI CEO KOD U JEDAN PROJEKAT ILI CPP fajl!

24.10.2017. godine sa predmeta