VISOKA POSLOVNA ŠKOLA
STRUKOVNIH STUDIJA
ČAČAK

SEMINARSKI RAD

Veliki brojevi

Mentor:_____ Student: _____
Profesor: _____ Br.Indeksa: _____

Sadržaj

Tekst zadatka:

Zadatak VB7: Veliki brojevi

Napisati program Veliki Brojevi koji omogućava rad sa celim brojevima sa velikim brojem cifara.

1. Napisati klasu Broj koja predstavlja jedan veliki broj
2. Napisati metode za: sabiranje, oduzimanje, množenje, celobrojno i realno deljenje, ostatak pri celobrojnom deljenju, odredjivanje NZD i NZS, korenovanje, stepenovanje, faktorijel i proveru da li je veliki broj prost.
3. Napisati program koji iz jedne sekvencijalne datoteke ucitava izraze (svaki sadrzi samo po jednu operaciju) a u drugu upisuje iste te izraze zajedno sa rezultatima primene operacija.

Modifikacije u odnosu na početni problem

- Veliki broj je opisan klasom Z (asocirajuci na oznaku za skup celih brojeva), od traženih metoda u klasu Z ugrađene su metode za: sabiranje, oduzimanje, množenje, celobrojno deljenje, ostatak pri celobrojnom deljenju i stepenovanje. Operacije faktorijel, celobrojno korenovanje, NZS i NZD kao i funkcija ProstBroj napisane su kao spoljne funkcije.
- Tražena operacija "Realno deljenje" pripada klasi Racionalnih brojeva (klasi razlomaka Q) a rezultat razlomljenog deljenja u decimalnom zapisu može se dobiti metodom Q::ispisFP(ostream& ostr , unsigned brojDecimala)
- Napisana je i funkcija Q sqrt(q,epsilon) - koja vraća približan broj kvadratnom korenu broja q (kao razlomak, koji se može prikazati decimalnom zapisu) koji odstupa od kvadratnog korena broja q najviše za epsilon.

Napomene

- Program je razvijan na GNU C++ pod linux OS-om i svi testovi izvršeni su kompajliranjem programa navedenim kompajlerom sa opcijom "-O" za optimizaciju brzine.
- Zbog upotrebe šablona u klasi bitfield nisam bio u mogućnosti da program podelim u više odvojeno kompajliranih modula. Sve klase su opisane u posebnim fajlovima jwork_bitfield, jwork_z i jwork_q sa ekstenzijama .cc i .h
- klase bitfield, Z i Q napisane su u prostoru imena "jwork"
- Sva vremenska testiranja koja su prikazana u opisu predstavljaju relativne podatke. Merenja data bez napomene računara na kome su izvedena, izvršena su na računaru sa 32-obitnim procesorom: Celeron 400mhz 128kb cache 800 bogomips

Kratak uvod i opis problema

Svi programski jezici sadrže takozvane ugrađene tipove podataka. U jezicima C i C++ to su char, short, int, long, float, double, i td.. sa eventualnim predznacima unsigned ili podrazumevanim signed. Navedeni tipovi podataka zauzimaju: char 8 bita, short 16 bita, int 32 bita, long 32 ili 64 bita. Svi ovi tipovi podataka mogu da registruju određen, konačan i relativno mali broj kombinacija koje se interpretiraju kao označeni ili neoznačeni brojevi. Na primer tip char može da sadrži 256 raznih kombinacija, tip short 65535 kombinacija, int 4294967296. Čak i ukoliko je procesor baziran na 64-obitnoj arhitekturi i kompajler podržava tip long sa 64-bita tada tip long može da sadrži najviše 18446744073709551616 razlicitih kombinacija odnosno brojeva ukoliko se govori o celim brojevima. Kada se govori o "Realnim brojevima" kod njih je situacija slična, opseg realnih brojeva je znatno veći (za tip double opseg se kreće do 10^308) ali broj razlicitih kombinacija tj različitih realnih brojeva je ograničen. To je postignuto proređivanjem brojeva dalekih od jedinice (prema +inf i prema nuli).

Veliki ceo broj npr : 123456789022233123145446700 3 može se u racunaru zapisati na razne načine, intuitivno prvo što pada napamet je da se broj zapiše kao niska karaktera char *s="123456789022233123145446700 3" odnosno niz tipa char tj "vector <char> s" (jer očigledno je da naveden primer premašuje kapacitet ugrađenog tipa unsigned int). Dobra strana ovog načina zapisa je jednostavnost unosa i ispisivanja broja dok je loša strana neefikasno trošenje memorije i spor rad sa aritmetičkim operacijama.

Veliki broj, ne može biti smešten u jednu mašinsku rec i potrebno ga je podeliti. Upravo je opisan najintuitivniji način deljenja tj svaka dekadna cifra predstavlja jedan karakter. Neki od razloga neefikasnosti takvog zapisa su gore navedeni. Ukoliko se, radi jednostavnosti rada i provere međurezultata pri konstrukciji samog programa, uzme da je veličina mašinske reci 8 bitova, najefikasnije sa stanovišta memorije je da se broj ne pamti u sistemu sa osnovom deset vec u sistemu sa osnovom 256. Odnosno u koliko se broj smešta u vector <unsigned int> tada je najoptimalnija osnova za rad 4294967296. Pošto je za čoveka, teže da koristi brojne sisteme sa osnovom većom od 16 (heksadekadni sistem) za prikaz sistema sa sa osnovom 2^n koristi se binarni sistem.

Najcešca operacija pri radu sa brojevima je operacija sabiranje i sve ostale operacije (- , *, / , %) se svode na sabiranje. Zbog toga je dobro iskoristiti sabirač samog procesora koji radi sa ugrađenim celobrojnim tipovima podataka. Zbog mogucih prenosa brojeva pri sabiranju nije maksimalno iskorišcena mašinska reč, jer joj je oduzet jedan bit koji se koristi kao kontrolni pri prenosu u sabiranju ili pri pomeranjima bitova.

Način na koji sam u ovom radu rešio problem velikih brojeva je opisivanje klase polja bitova (bitfield). Objekat iz klasae polje bitova dinamički alocira potreban broj bajtova za zadato polje koji se kasnije može smanjivati ili povecavati. Veličina polja bitova nije ograničena osim od strane maksimalne količine memorije koja se može alocirati. Uglavnom zbog testiranja, klasa bitfield napisana je kao šablon sa parametrom “tip reci” koji ce biti korišcen. Do najefikasnijeg rešavanja problema velikih brojeva dolazi se korišćenjem reči koja zauzima veličinu registra (long int). Cela efikasnost rada sa velikim brojevima bazira se na klasi bitfield i upravo zbog toga u klasi nisu korišćeni izuzeci

Kratak opis šablon klase bitfield.

Opis pojmova:

- polje ili polje bitova - predstavlja ceo objekat klase bitfield
- bajt ili reč - opisuje tip podataka od kojih je sačinjeno polje
- bit - element polja bitova
- kontrolni bit - bit koji se koristi za prenos i sabiranje

Šematski (ako se kao argument šablona koristi tip unsigned char) i ako se govori o polju sa 28 bitova, to bi bilo prikazano ovako:

```
<0>0000000 | <0>0000000 | <0>0000000 | <0>0000000              polje bitova
 ^           ^           ^           ^                        kontrolni bitovi
 - -------    - -------    - -------    - -------             bajtovi
      ^                                                      jedan od bitova
   ^                                                         bit najvece tezine
                                              ^              bit najmanje tezine
```

Zbog namene objekata klase bitfield (rad sa velikim brojevima) uveden je kontrolni bit koji predstavlja bit najveće težine u bajtu, da bi bilo moguće iskoristiti ugrađene operatore za rad sa rečima. Na primer operator pomeranja na levo koji pomera sve bitove jednog bajta (operator <<) koji je implementiran na nivou procesora, kao i operator inkrementacije, sabiranja. itd... Pomeranje na levo u polju bitova vrši se na taj način što se na levo stranu pomeraju bitovi pojedinačnih bajtova. Ukoliko dođe do popune kontrolnog bita odgovarajućeg bajta u narednom bajtu se posle pomeranja dodaje bit na mestu najmanje težine u tom bajtu. Tako se u gore navedenom primeru sa 28-bitnim poljem čija je osnova tip char, operacija pomeranja na levo sprovodi u četiri osnovna koraka (dok, ako bi se pomeranje vršilo bit po bit, bilo bi potrebno izvršiti 28 koraka). Sa druge strane ako bi se umesto tipa char koristio tip short pomeranje bi se odvilo u dva ciklusa odnosno ako bi se koristio tip int bio bi potreban samo jedan ciklus. Problem prenosa izgubljenih bitova može se rešiti i na drugi način ali kako su kontrolni bitovi neophodni za druge operacije isti su iskorišćeni i za pomeranje.

Na klasi polje bitova implementirani su metodi za pristup pojedinačnim bitovima, upis, čitanje, pomeranje bitova u polju, logičke operacije (and,or,xor,not), poređenje sa drugim poljem i sl...

Kratak opis klase Z.

Klasa Z predstavlja nadgradnju i jednu od konkretizacija polja bitova, koje se može koristiti (uz eventualne modifikacije) i za neke druge probleme. Kao takva klasa Z nasleđuje klasu bitfield, preopterećuje neke od operatora koji su već definisani u polju bitova, uvodi nove operatore (poređenje uređenog skupa celih brojeva), uvodi operacije sabiranja, množenja, celobrojnog deljenja kao i ostatka pri celobrojnom deljenju i ispis celog broja u osnovi deset.

Ceo broj je zapisan u potpunom komplementu i za informaciju o znaku broja koristi se bit najveće težine u polju bitova tzv signBit

Ključni metod u klasi Z je metod **addEqSize** koji se koristi za sabiranje i indirektno za oduzimanje, množenje, deljenje i računanje ostatka. Algoritam za sabiranje bazira se na sabiranju dva broja u potpunom komplementu i koristi operator += za ugrađene tipove podataka.Tako da se sabiranje dva broja ne sabira bit po bit već bajt po bajt. U sabiranju dva ugrađena tipa podataka npr dva tipa **unsigned char**, zbir brojeva 150+150 biće 44 zbog sabiranja po modulu, ali pošto se od bajta koristi n-1 bit (zbog kontrolnog bita) do ovoga ne može doći tj popuniće se polje u kontrolnom bitu, ali to je samo znak da postoji prenos jedne binarne jedinice u zbir dva sledeća bajta.

koriscenje klase string iz standardne biblioteke klasa

Zbog jednostavnosti koriscenja klase string ona je korišćenja na manje frekventnim mestima, postoje metodi i funkcije koje rade sa niskama i ponavljaju se više puta u kratkom vremenu, za koje se uspostavlja da je neefikasno koristiti objekat klase string i iz tog razloga su te metode napisane i korišćenjem pokazivača na karaktere. koji će se način prevesti zavisi od direktive **CONVERSION_USE_C_STYLE.** Ukoliko je direktiva definisana koristiće se niske c-ovskog tipa a u suprotnom koristiće se klasa string.

primer: konvertovanja broja koji u dekadnoj reprezentaciji ima 10,000 cifara u binarni pa zatim oped u dekadni

funkcija	vreme sa niskama	vreme sa stringom
dec2bin	23.19 sec	32.85 sec
bin2dec	25.92 sec	99.02 sec

Načini konstrukcije objekta iz klase Z (sa nekim primerima)

konstruktor	vrednost broja
Z(325)	325
Z("123456789012345567")	123456789012345567
Z("77765e30")	77765000000000000000000000000000000
Z("77765e-30")	0
Z("77765e-3")	77
Z("-512321321312")	-512321321312
Z("e15")	1000000000000000
Z("e-15")	0

Pošto je dopušteona konstrukcija polja preko objekta klase string, može doći do grešaka u ulazu tj prosleđivanja neregularnih brojeva, zadatak ovog rada nije bio rad sa regularnim izrazima te su obrađeni samo neki najčešći slučajevi grešaka. Tako da će pri konstrukciji sledećih celih brojeva biti ispaljen izuzetak

Konstruktor	Razlog
Z("85s9")	slovo u broju (razlčito od slova e ili E)
Z("e15e2")	eksponent u eksponentu (nije dozvoljeno)
Z("321-21)	znak može biti samo na početku ili iza slova e/E

Kratak opis klase Q.

Klasom razlomaka Q može se predstaviti razlomljen broj. Daleko od toga da se može predstaviti svaki broj između neka dva cela broja kao npr algebarski iracionalni ili transcedentni brojevi ali zahvaljujući tome da se objektom iz klase Z može prikazati proizvoljno veliki ceo broj i da je skup **Q** gust na skupu **R**, objektom iz klase Q se može proizvoljno približiti bilo kom realnom broju. Ključni metod u klasi Q je **skrati** koji dovodi brojilac i imenilac u razlomku na dva uzajamno prosta broja. Ovaj metod koristi funkciju nzd(Z,Z) za cele brojeve. Ukoliko bi se izbeglo dovođenje brojioca i imenioca razlomka na dva uzajamno prosta broja pri radu sa racionalnim brojevima objekat tipa Q bi višestruko brže rastao pri, na primer, sabiranju brojeva.

primer: **s=sum(0,50,1/(n!))** sa funkcijom za skracivanje vreme izvrsavanja 1.0176 sec, potrebna memorija 1.06 Kb, dok bez funkcije za skracivanje vreme izvrsavanja 39.7804 sec, potrebna memrija 45.44 Kb

Načini konstrukcije objekta iz klase Q (sa nekim primerima)

konstruktor	razlomak	decimalan prikaz broja
Q(1,3)	[1 / 3]	0.(3)
Q(15,-5)	[-3]	-3
Q(-2)	[-2]	-2
Q("123.456")	[15432 / 125]	123.456
Q("0.553")	[553 / 1000]	0.553
Q("0.32e5")	[32000]	32000
Q(".15e-5")	[3 / 2000000]	0.0000015
Q("e-5")	[1 / 100000]	0.00001
Q("-.25(15)")	[-83 / 330]	-0.2(51)
Q("15.321(0005)")	[38298671 / 2499750]	15.321(0005)
Q("15/4")	[15 / 4]	3.75
Q("7e3/3")	[7000 / 3]	2333.(3)
Q("15/e-5")	[1500000]	1500000
Q("15.25(5)/3")	[1373 / 270]	5.0(851)
Q("/35")	[1 / 35]	0.0(285714)

Broj zadat i ispisan u zagradama predstavlja broj koji se periodično ponavlja, npr u primeru Q("'-.25(15)") periodično se ponavlja broj 15 tj zapis broja bi bio -0.251515151515... što se može protumačiti i kao "-0.2(51)"

Greške pri konstrukciji broja Q preko stringa takođe mogu izazvati izuzetke pri nekim greškama, kao npr:

konstruktor	vrednost broja
Q("7j3")	slovo u broju (koje nije slovo e)
Q("6.55e16e25")	eksponent sadrzi eksponent (nije dozvoljeno)
Q("123(5).321")	periodicno ponavljanje moze biti samo u razlomljenom delu broja
Q("32.221()")	izostavljen periodican deo
Q("-12.34(66e3)")	eksponent u periodicnom delu (nije dozvoljeno)
Q("1.234e-5(567)")	period ne moze biti zadat posle eksponenta
Q("7.21(553")	nije zatvorena zagrada za oznaku perioda
Q("3/0")	deljenje nulom nije dozvoljeno

Detaljan opis klasa Bitfield, Z i Q

template <class T>
class bitfield

naziv i parametri metoda	**kratak opis**
konstrukcija i inicijalizacija objekta	
bitfield()	podrazumevani konstruktor
bitfield(T)	konstrukcija brojem tipa T
bitfield(unsigned bits,T pattern)	konstrukcija preko zadatog broja bitova i patterna
bitfield(const char *s)	konstrukcija preko niske znakova
bitfield(string s)	konstrukcija preko objekta iz klase string
void init(unsigned nBits,T pattern)	inicijalizacija
void init(const char *s)	
void init(T value)	
const metodi (metodi koji daju informacije)	
unsigned size()	broj bitova u polju
unsigned sizeInWords()	broj reči (bajtova) u polju
unsigned whichWord(unsigned pos)	reč u kojoj se nalazi bit sa pozicije pos
unsigned whichBit(unsigned pos)	pozicija bita pos u bajtu iz polja bitova
bool test(unsigned pos)	vraća stanje bita (1 ili 0)
bool test_(unsigned pos)	stanje bita, uključujući i kontrolne bitove
bool any()	da li je uključen bilo koji bit
bool none()	da li nema uključenih bitova
unsigned count()	vraća broj uključenih bitova
bool lastBit()	da li je uključen poslednji bit
bool firstBit()	da li je uključen prvi bit
T getWord(unsigned pos)	vraća reč sa tražene pozicije
bool CtlTest(unsigned word)	vraća sadržaj kontrolnog bita sa
bool NotEqualWith(const bitfield<T>&b)	vraća true ukoliko je objekat različit od objekta b
bool EqualWith(const bitfield <T>&b)	vraća true ukoliko je objekat isti kao objekat b
bool isEqual(const bitfield <T>&)	poredi polja iste dužine
bool operator==(const bitfield <T>&)	overloading operatora jednakosti
bool operator!=(const bitfield <T>&)	overloading operatora nejednakosti
bool operator[] (unsigned pos)	overloading operatora indeksnog pristupa (za čitanje)
write_binary_CTL(ostream& ostr)	ispis poljabitova (sa označenim kontrolnim bitovima)
write_binary(ostream& ostr)	ispis poljabitova
metodi koji menjaju sadržaj klase	
void extend(unsigned size, T pattern)	proširivanje polja bitova
void cut(unsigned size=1)	sečenje polja bitova
void flip()	promena vrednosti svim bitovima (negacija polja)
void flip(unsigned pos)	promena vrednosti bitu lokaciji pos
void set()	uključivanje svih bitova
void set(unsigned pos)	uključivanje bita na poziciji pos
void reset()	isključivanje svih bitova
void reset(unsigned pos)	isključivanje bita na poziciji pos
void setBit(unsigned pos, bool bit)	postavljanje vrednosti bita na poziciji
void setWord(unsigned, T)	postavljanje reči na poziciji

void plusWord(unsigned, T)	dodavanje reči na već postojeću reč
void CtlSet(unsigned word)	uključivanje kontrolnog bita
void CtlReset(unsigned word)	isključivanje kontrolnog bita
void CtlFlip(unsigned word)	menjanje vrednosti kontrolnom bitu
void CtlReset()	isključivanje svih kontrolnih bitova
void do_and(const bitfield <T> &b)	logičko "I" sa poljem bitova objekta b
void do_or(const bitfield <T> &)	logičko "ILI" sa poljem bitova objekta b
void do_xor(const bitfield <T> &)	logičko "isključivo ILI" sa poljem bitova objekta b
bool do_left1()	pomeranje bitova u objektu za jedan bit na levo
bool do_left1streach()	pomeranje bitova na levo uz proširenje polja
bool do_left1circle()	ciklično pomeranje bitova na levo
bool do_right1()	pomeranje na desno (za jedan bit)
bool do_right1streach()	uz raširenje polja ukoliko je to potrebno
bool do_right1circle()	ciklično pomeranje na desno
bool do_inc()	inkrementacija
bool do_dec()	dekrementacija
reference operator[](unsigned pos)	overloading operatora za indeksni pristup (upis bitova)
bitfield<T>& operator<<=(unsigned n)	overloading operatora za pomeranje na levo
bitfield<T>& operator>>=(unsigned n)	overloading operatora za pomeranje na desno
operator&=(const bitfield <T>&)	AND
operator|=(const bitfield <T>&)	OR
operator^=(const bitfield <T>&)	XOR

class Z (objekti klase Z nasledjuju objekte klase bitfield<long>)

naziv i parametri metoda	kratak opis
konstrukcija	
Z()	podrazumevani konstruktor
Z(signed wordBase)	konstruktor preko označenog ugrađenog tipa
Z(const string &)	konstruktor preko objekta iz klase string
const metodi (metodi koji daju informacije)	
Z makeZfromStr(const string & str)	kreira ceo broj preko niske
bool signBit()	vraća informaciju o znaku broja
bool positive()	**true** ako je objekat pozitivan broj
bool negative()	**true** ako je objekat negativan broj
bool isNull()	**true** ako je objekat 0 broj (nula)
bool veciEqSizeEqSign(const Z& b)	poredjenje sa brojem b (objekti su iste dužine u bitovima)
bool manjiEqSizeEqSign(const Z& b)	poredjenje sa brojem b (objekti su iste dužine u bitovima)
bool veci_jednakoEqSizeEqSign (const Z& b)	poredjenje sa brojem b (objekti su iste dužine u bitovima)
bool manji_jednakoEqSizeEqSign (const Z& b)	poredjenje sa brojem b (objekti su iste dužine u bitovima)
bool veci_sgn(const Z&)	veći po znaku
bool manji_sgn(const Z&)	veći po znaku
bool operator>(const Z& b)	overloading operatora veće
bool operator>=(const Z& b)	overloading operatora veće i jednako
bool operator<(const Z& b)	overloading operatora manje
bool operator<=(const Z& b)	overloading operatora manje i jednako
bool operator!=(const Z& b)	overloading operatora različito
bool operator==(const Z& b)	overloading operatora jednako
string bin2dec()	konvertovanje binarnog broja u string
operator string()	konverzija u string
metodi koji menjaju sadržaj klase	
void inv()	menja znak broju (potpun komplement binarnog broja)
bool do_right1holdSignBit()	pomeranje na desno sa zadrškom znaka za bit
void addEqSize(const Z &)	ddavanje drugog broja (iste veličine u bitovima)
Z& mul2()	množenje brojem 2
Z& div2()	deljenje brojem 2
Z& mul2poz()	množenje sa 2 (samo za pozitivne brojeve)
Z& div2poz()	delenje sa 2 (samo za pozitivne brojeve)
bool abs()	pretvara broj u pozitivan
Z& mul_poz(Z)	množi objekat brojem tipa Z
void divMod(Z a, Z b, Z& div, Z& mod)	deli dva broja, vraća količnik i ostatak pri delenju
void srediZRekurzivno()	sređuje polje bitova (minimizira ga ukoliko je došlo do nagomilavanja nepotrebnih nula bitova na mestima na kojima oni nemaju nikakvo značenje u prikazu broja
void srediZ()	isključuje kontrolne bitove i poziva srediZRekurzivno
Z& pow(Z on)	stepenovanje brojem iz klase Z
Z& pow(int on)	stepenovanje brojem int (brža i češće korišćena varijanta)
Z& pow2()	kvadriranje
Z& operator ++(int)	overloading sufiksne inkrementacije
Z& operator ++()	overloading prefiksne inkrementacije
Z& operator --(int)	overloading sufiksne dekrementacije
Z& operator --()	overloading prefiksne inkrementacije
Z& operator-()	overloading operatora za unarni minus
Z& operator+=(const Z&)	dodavanje broja iz klase Z
Z& operator-=(Z)	oduzimanje broja iz klase Z
Z& operator *= (Z)	množenje brojem iz klase Z
Z& operator/=(const Z&)	celobrojno deljenje brojem iz klase Z
Z& operator%=(const Z&)	računanje ostatka pri delenju sa brojem iz klase Z
void dec2bin(const string &)	konvertovanje broja iz dekadne reprezentacije u binarnu
funkcije	
Z pow(Z z, Z on)	stepenovanje

Z pow(Z z, int on)	int verzija za stepenovanje
Z pow2(Z z)	kvadriranje
Z nzd(const Z& a, const Z& b)	nalaženje najvećeg zajedničkog delitenja
Z nzs(const Z& a, const Z& b)	nalaženje najmanjeg zajedničkog sadržalaca
Z operator/(Z a,const Z& b)	operator delenja
Z operator%(Z a,const Z& b)	operator ostatka
Z operator+(Z a,const Z& b)	operator sabiranja
Z operator*(Z a,const Z& b)	operator množenja
Z operator-(Z a,const Z& b)	operator oduzimanja
Z factoriel(Z)	rekurzivno računanje faktorijela
Z exp10(const Z & exp)	računanje broja 10^exp
Z rz_sqrt(const Z& x, Z a, Z b)	rekurzivno računanje kvadratnog korena za ceo broj (metodom polovljena intervala)
Z sqrt(const Z & x)	vraća celobrojan koren celog broja
bool prostBroj(const Z& z)	vraca **true** ako je broj prost

class Q (u objekat klase Q agregirana su dva objekta iz klase Z)

naziv i parametri metoda	kratak opis
konstrukcija	
Q(const Z& _a=0,const Z& _b=1);	konstruktor preko celih brojeva (igra i ulogu podrazumevanog konstruktora)
Q(int)	konstruktor preko broja tipa integer
Q(double)	konstruktor preko broja tipa double
Q(string)	konstruktor preko niza znakova
const metodi (metodi koji daju informacije)	
Z get_a()	vrednost imenioca
Z get_b()	vrednost brojioca
operator string()	konvertuje racionalan broj u string
operator Z()	konvertuje racionalan broj u ceo broj
signed short sign()	vraća znak broja (-1 za negativan, 1 za pozitivan)
bool positive()	**true** ako je objekat pozitivan broj
bool negative()	**true** ako je objekat negativan broj
bool operator == (const Q&)	overloading za operator jednakost
bool operator != (const Q&)	overloading za operator nejednakost
bool operator < (const Q&)	overloading za operator
bool operator <= (const Q&)	overloading za operator
bool operator > (const Q&)	overloading za operator
bool operator >= (const Q&)	overloading za operator
void ispis(ostream& ostr)	ispis u obliku razlomka
void ispisFP(ostream& ostr,unsigned)	ispis u obliku broja sa pokretnim zarezom (sa zadatim brojem decimala ako ih ima)
void ispisFPP(ostream& ostr)	ispis u obliku decimalnog broja sa svim decimalama, gde se periodican deo ispisuje samo jednom u zagradi)
metodi koji menjaju sadržaj klase	
void decimalan(const string & s)	koristi se za konstrukciju broja preko decimalnog zapisa iz stringa
void razlomak(const string & s, int i)	koristi se za konstrukciju broja zadata stringom koji može da sadrži dva decimalna ili cela broja razdvojena "/"
void skrati()	skraćuje brojilac i imenilac tako da razlomci budu uzajamno prosti
void inv()	inverzija racionalnog broja (menja brojilac i imenilac)
void neg()	menja znak racionalnom broju
bool abs()	konvertuje racionalan broj u pozitivan, vraća true ukoliko je došlo do konverzije
Q& pow(Z on)	stepenovanje racionalnog broja celim brojem
Q& pow(int on)	stepenovanje int brojem
Q& pow2()	kvadriranje
Q& operator ++(int)	overloading sufiksne inkrementacije
Q& operator ++()	overloading prefiksne inkrementacije
Q& operator --(int)	overloading sufiksne dekrementacije
Q& operator --()	overloading prefiksne inkrementacije
Q& operator-()	overloading operatora za unarni minus
Q& operator+=(const Z&)	dodavanje broja iz klase Z
Q& operator-=(Z)	oduzimanje broja iz klase Z
Q& operator *= (Z)	množenje brojem iz klase Z
Q& operator/=(const Z&)	celobrojno deljenje brojem iz klase Z
funkcije	
Q abs(Q)	apsolutna vrednost racionalnog broja
Q operator + (Q a,const Q& b)	
Q operator - (Q a,const Q& b)	
Q operator * (Q a,const Q& b)	
Q operator / (Q a,const Q& b)	
Q rq_sqrt(const Q& a, Q x, const Q& epsilon)	rekurzivna funkcija za korenovanje
Q sqrt(const Q & x, const Q & epsilon)	koren racionalnog broja sa zadatom tačnošću
Q pow(Q q, Z on)	stepenovanje celim brojem
Q pow(Q q, int on)	stepenovanje integer brojem

Q pow2(Q q)	kvadriranje
razdvoji(const string & s, int & znak, Z & ceo, string & razlomljen, string & periodican, Z & eksponent)	koristi se pri konstrukciji broja preko stringa i razdvaja delove broja iz stringa
bool checkDec(const string & s)	proverava korektnost stringa koji predstavlja decimalni zapis broja
int findChar(const string & str, char c, int & repeat)	traži karakter u stringu, i vraća njegovu poziciju prvog pojavljivanja kao i broj pojavljivanja, dok ako karakter ne postoji tada vraća -1

Struktura programa, način kompajliranja i primeri upotrebe programa

potrebni fajlovi:

broj.h	header fajl koji povlaci sve ostale, dovoljno je uključiti samo njega
jwork_define.h	definicije nekih konstanti i opcija
jwork_bitfield.h	dekleracija klase bitfield
jwork_z.h	dekleracija klase Z
jwork_q.h	dekleracija klase Q
jwork_errorMsgs.cc	poruke pri ispaljivanju izuzetaka
jwork_c_style_conversion.cc	funkcije dec2bin i bin2dec korišćenjem niski C-ovskog tipa
jwork_string_style_conversion.cc	funkcije dec2bin i bin2dec korišćenjem objekata klase string
timetest.cc	funkcije za merenje vremena
jwork_bitfield.cc	šablon klasa bitfield
jwork_z.cc	klasa Z
jwork_q.cc	klasa Q
calc.cpp	program

Program je uspešno preveden na GNU C++ prevodiocu i Borland builderu verzija 5, za prevođenje programa pod linuxom potrebno je napisati **g++ calc.cpp -O -o calc** dok je za kompajliranje u borland builderu potrebno otvoriti fajl **calc.bpr** i pokrenuti kompajler

argumenti programa **calc.exe** odnosno calc (pod linuxom) su: opcije programa, ime ulazne datoteke i ime izlazne datoteke. ime izlazne datoteke ne mora biti navedeno i tada će se izlaz preusmeriti na standardan izlaz (cout), ukoliko se navede opcija **-console** ulaz se očekuje preko tastature i izlaz se izdaje na standardnom izlazu, opcija **-timetest** će ispisati vreme potrebno za izvođenje operacije. opcija **-q** će tretirati ulazne podatke kao racionalne brojeve, u suprotnom oni se tretiraju kao celi brojevi

primeri:

test.dat
+ 700000000003 250

calc.exe test.dat
rezultat operacije 700000000003+250=700000000253

test.dat
* 70e5 253

calc.exe test.dat
rezultat operacije 7000000*253=1771000000

test.dat
/ 950 34

calc.exe test.dat
rezultat operacije 950/34=27

ukoliko se program pozove opcijom **-q**
calc.exe test.dat -q

rezultat operacije [950]/[34]=[475 / 17]
decimalan zapis sa 50 decimala: 27.94117647058823529411764705882352941176470588235294

test.dat
factoriel 50

calc.exe test.dat
faktorijel 50=30414093201713378043612608166064768844377641568960512000000000000

test.dat
sqrt 50

calc.exe test.dat
kvadratni koren broja 50 je 7

test.dat
sqrt 50 E-50

calc.exe test.dat -q
kvadratni koren broja [50]
decimalan zapis sa 50 decimala: 7.07106781186547524400844362104849039284835937688474

Podaci nekoliko merenja preciznosti i brzine izvšavanja nekih operacija

Na samom početku bilo je reči o gustini zapisa brojeva tipom double, koji je odozgo ograničen brojem 10^308, Međutim gustina zapisa nekad može biti od značaja (kada se radi sa mikro ili makro veličinama, npr obrada nekih astronomskih podataka).. kroz par primera prikazana su odstupanja podataka zapisanih tipom Q, double i float (kod tipa Q nema odstupanja)

broj 3e10
Q: 30000000000
double: 30000000000
float: 30000001024

3.15e30
Q: 3150000000000000000000000000000
double: 3150000000000000020415321735168
float: 3150000085178450455568731602944

3e35
Q: 300000000000000000000000000000000000
double: 299999999999999990590098323753926656
float: 299999992468502999322841559739662336

Vremenski testovi nekih operacija izvršeni su na tri računara (ALAS,ALFA i BETA) sledećih karakteristika:

ime: ALAS
adresa: alas.matf.bg.ac.yu
racunar: 2 x (Pentium III (Coppermine), CPU 790Mhz, 256Kb Cache, 1576 bogomips)

ime: ALFA
adresa: alfa.mas.bg.ac.yu
racunar: (Pentium III (Coppermine), CPU 994Mhz, 256Kb Cache, 1979 bogomips)

ime: BETA
adresa: beta.mas.bg.ac.yu
racunar: (Alpha EV56, CPU 400 Hz, 369 bogomips)

Napomena! rezultati testiranja su okvirni, +/- 5% zbog zauzetosti procesora u trenutku testiranja

test rada sa celim brojevima

- računanje faktorijela

```
1000!
```
ALAS: 0.067918 sec ALFA: 0.054801 sec BETA: 0.098761 sec

* rezultat ovog testiranja je broj koji u dekadnom zapisu ima 2568 cifara

```
1000!
```
ALAS: 10.8418 sec ALFA: 8.83440 sec BETA: 14.4386 sec

* rezultat ovog testiranja je broj sa 35660 cifara u dekadnom zapisu

- provera da li je broj prost

```
prosejavanje prostih brojeva u prvih 10000 prirodnih brojeva ( ima ukupno 1229 prostih brojeva)
```
ALAS: 4.53969 sec ALFA: 3.63152 sec BETA: 14.0091 sec

test rada sa racionalnim brojevima

- sumiranje razlomaka

```
e300 = sum ( n=[0,300], 1/n! )
```
ALAS: 14.7862 sec ALFA: 12.3354 sec BETA: 26.0939 sec

- iterativno nalaženje korena

```
Racunanje sqrt(2) sa tacnoscu 1*10^-1000
```
ALAS: 6.35768 sec ALFA: 5.24278 sec BETA: 10.6752 sec