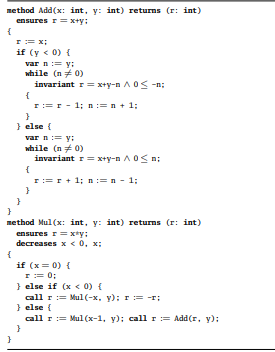
# Primeri jednostavnih Dafni implemmentacija

Sada ćemo predstaviti uspešnost Dafni verifikatora na nekoliko uobičajenih problema.

## Sabiranje I množenje brojeva

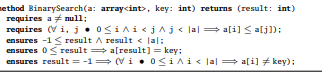
Ovde ćemo verifikovati program koji vrši sabiranje 2 broja tako što vrši inkrementaciju. Verifikujemo i algoritam množenja 2 broja , množenje vršimo tako što dodajemo prvi operand neophodan broj puta, to jest koristimo gore naveden algoritam sabiranja. Sabiranje će biti iterativno a množenje rekurzivno.



U oba algortima postuslov je verifikovan, a to je I završetak metoda. U sabiranju brojeva vidimo da smo iskoristili I invarijantu petlje.

## Binarna pretraga niza

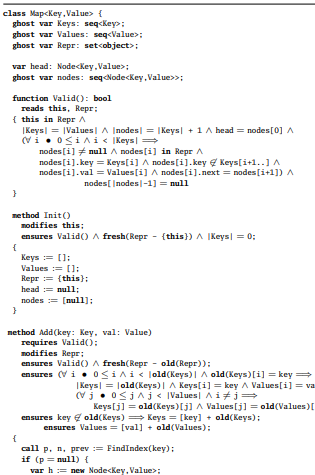
Ovde ćemo predstaviti verifikaciju ovog dobro poznatog algoritma koji u sortiranom nizu brojeva nalazi tražen broj u linearnoj složenosti. Pored samog algoritma binarne predrage u Dafniju moramo verifikovati I ulaz, to jest da su svi brojevi u sortiranom poretku.

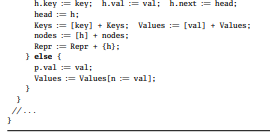




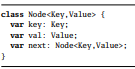
## Implementacija mape

U ovom primeru ćemo verifikovati generičku klasu Map, koja nije deo Dafnija već smo je implementirali. Ona je predstavljena povezanom sekvencom ključeva I vrednosti, gde ključ k na poziciji I u sekvenci ključeva ima odgovarajuću vrednost v koje je takođe na poziciji I u sekvenci vrednosti. Generička mapa je implementirana povezanom listom čvorova gde svaki čvor sadrži ključ, vrednost I pokazivač na sledeći čvor.





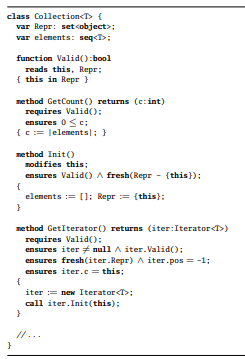
Sada ćemo prikazazi kako izgleda ćvor mape, to jest klasu Node.

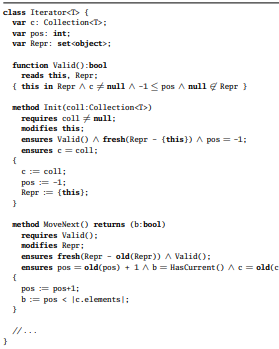


Iteratori

Ovde ćemo verifikovati program koji koristi iteratore za neki kolekcijski tip. Za ovaj problem neophodno je bilo implementirati I klasu Iterator I klasu Collection. Kolekcijska klasa je implementirana kao sekvenca generičkih tipova sa metodama za inicijalizovanje kolekcije, vraćanje vrednosti, dodavanje vrednosti I dohvatanje iteratora iz kolekcije.

Program skladišti elemente kolekcije sekvencionalno i proverava da li iterator vraća ispravan element. Takođe proveravamo da iterator ne uništi kolekciju preko koje iterira.





# Rezultati

Vremena izvršavanja verifikacije su redom : 3,5, 3,7, 4,9 I 3,9 sekundi. Rezultati su vrlo zadovoljavajući. Jedan veliki problem u nasim implemenacijama je to što Dafni koristi matematičke intidžere koje većina programskih ne koristi. Tako da u Dafni programima ne postoji mogućnost prekoračenja što je inače vrlo realan problem.

# Zaključak

Dafni je jedan od alata koji se trenutno koristi za u akademskim I industrijskim krugovima za konstrukciju pouzdanog softvera. Sa jedne strane Dafni je lak za učenje zato što sadrži većinu funkcionalnosti koje sadrže I najpopularniji programski jezici današnjice. Sa druge strane Dafni sadrži dokaze koji moraju da se ispune kao deo programskog koda, što umnogome olakšava upotrebu i povećava čitljivost programa, jer baš oni objašnjavaju korektnost algoritma. Dafni se trenutno ne koristi dovoljno ni u industriji ni u teorijskim okvirima ali imajući u vidu da je ovo relativno novi alat koji drugi programski jezici ne podržavaju dovoljno, može se u budućnosti očekivati veća upotreba oog alata.