RAČUNALNIŠTVO I
1997/98

4. domača naloga

19. maj 1998

Kot osnova za vaše delo morajo obvezno biti moduli iz prejšnjih domačih nalog, ali tisti, ki smo jih skupaj sestavili na vajah oziroma predavanjih. Te module najdete v našem WWW sestavku ali na področju

`Z:\Predavanja\Polde` .

1. Na vajah smo napisali program, ki je opravljal funkcije kalkulatorja. Pri tem je uporabljal za svoje delo sklad celih števil.

   NALOGA: *Napišite enak program, le da bo omogočal računanje s celimi števili poljubne natančnosti. Pri tem (lahko) uporabite modula* ***Cint*** *iz prejšnje domače naloge in* ***Skladi*** *(modul z vzorci) z vaj.*

   Program naj uporablja enake ukaze kot smo jih definirali na vajah:

   V<število> prebere število in ga naloži na sklad

   I vzame število s sklada in ga izpiše

   S vzame s sklada dve števili, ju sešteje in vsoto naloži ponovno na sklad

   O vzame s sklada prvo število, vzame s sklada drugo število, odšteje prvo število od drugega in razliko ponovno naloži na sklad

   Z vzame s sklada dve števili, ju zmnoži in zmnožek naloži ponovno na sklad

   K vzame s sklada prvo število, vzame s sklada drugo število, zdeli prvo število z drugim in kvocient ponovno naloži na sklad

   D vzame s sklada prvo število, vzame s sklada drugo število, zdeli prvo število z drugim in ostanek ponovno naloži na sklad

2. V zadnji domači nalogi ste morali napisati definicijski modul za računanje s poljubno natančnimi ulomki (Crac):

```
#include "../../code/Cint/Cint.h"      // vecnatancnostna cela stevila

class Crac {
/* Desc:    Razred racionalnih stevil za vec-natancnostno racunanje. Razred
            omogoca spremembo predstavitve iz long in iz niza (string,
            char *). V predstavitvi kot niz ima obliko (brez
            narekovajev): "<predznak><stevke>/<stevke>". Predznak je lahko
            "+", "-", ali "" (nic). V slednjem primeru se privzame, da je
```

```
            predznak "+" -- to je, stevilo je nenegativno. Razred
            omogoca tudi spremembo predstavitve nazaj v niz.
*/

//-----------------------------------------------------------------
//------------------------------------------[ ZASEBNO, NE SKILI!! ]
private:
  Cint mStevec;                // stevec ulomka in ...
  Cint mImenovalec             // ... njegov imenovalec

//-----------------------------------------------------------------
//--------------------------------------------------------[ JAVNO ]
public:

// - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
// - - - - - - - - - - - - - - - - - - - -[ ustvarjanje / unicevanje ]
  Crac();
/* Desc:   Konstruktor. Ustvari novo stevilo z vrednostjo nic.
   Parms:  -
   Result:
   Pre:
   Post:   this = 0
*/

// - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
  Crac(char * stevec, char * imenovalec);
  Crac(long stevec,   long imenovalec);
  Crac(Cint stevec,   Cint imenovalec);
/* Desc:   Konstruktor. Ustvari novo stevilo z vrednostjo
           stevec/imenovalec. Ce vrednosti stev ne moremo prirediti ima
           predmet vrednost 0.
   Parms:  stevec -- prepisovani stevec (po vrednosti)
           imenovalec -- prepisovani imenovalec (po vrednosti)
   Result:
   Pre:
   Post:   (this = stevec/imenovalec) OR (this = 0)
*/

// - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
  bool Priredi(char * stevec, char * imenovalec);
  bool Priredi(long stevec,   long imenovalec);
  bool Priredi(Cint stevec,   Cint imenovalec);
  bool Priredi(Crac stevilo);
/* Desc:   Stevilo, ki ga predstavlja stevec/imenovalec (odnosno stevilo),
           prepisemo v ta predmet. Ce je predmet pred tem hranil neko
           drugo vrednost, slednjo pozabimo. Ce prepisa ne moremo izvesti,
           vrnemo false.
   Parms:  stevec -- prepisovani stevilec (po vrednosti)
           imenovalec -- prepisovani imenovalec (po vrednosti)
           stevilo -- prepisovani stevilo (po vrednosti)
   Result: RESULT
   Pre:
   Post:   IF (RESULT) THEN this = stevece/imenovalec (ali stevilo) ENDIF
*/
```

```
// - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
  ~Cint();
/* Desc:   Destruktor.
   Parms:  -
   Result:
   Pre:
   Post:
*/

// - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
// - - - - - - - - - - - - - - - - - - - - - - - [ matematicne operacije ]
  bool Pristej(Crac stev);
/* Desc:   Pristejemo stevilo stev. Ce pristevanja ne moremo izvesti,
           naznacimo to z vrnitvijo vrednosti false.
   Parms:  stev -- pristevatelj (po vrednosti)
   Result: RESULT
           ( Ce rezultat ustreza pogojem, funkcija vrne T, sicer F. )
   Pre:
   Post:   IF RESULT THEN this'= this + stev
           ELSE this' = this ENDIF
*/

// - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
  bool Odstej(Crac stev);
/* Desc:   Odstejemo stevilo stev. Ce odstevanja ne moremo izvesti,
           naznacimo to z vrnitvijo vrednosti false.
   Parms:  stev -- odstevatelj (po vrednosti)
   Result: RESULT
           ( Ker rezultat vedno ustreza pogojem, nam funkcija vrne T. )
   Pre:
   Post:   IF RESULT THEN this'= this - stev
           ELSE this' = this ENDIF
*/

// - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
  bool Zmnozi(Crac stev);
/* Desc:   Stevilo, ki ga predstavlja ta predmet, zmnozimo s stev. Ce
           mnozenja ne moremo izvesti, naznacimo to z vrnitvijo
           vrednosti false.
   Parms:  stev -- faktor (po vrednosti)
   Result: RESULT
           ( Ce rezultat ustreza pogojem, funkcija vrne T, sicer F. )
   Pre:
   Post:   IF RESULT THEN this'= this * stev
           ELSE this' = this ENDIF
*/

// - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
  bool Zdeli(Crac stev);
/* Desc:   Stevilo, ki ga predstavlja ta predmet, zdelimo s stev.
           Ce deljenja ne moremo izvesti (recimo stev = 0),
           naznacimo to z vrnitvijo vrednosti false.
   Parms:  stev -- delitelj (po vrednosti)
```

```
   Result: RESULT
   Pre:
   Post:   IF RESULT THEN (this= this' * stev)
           ELSE this' = this ENDIF
*/

// - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
// - - - - - - - - - - - - - - - - - - - - - - - - - [ poizvedovanja ]
  char * Vrednost();
/* Desc:   Stevilo, ki ga predstavlja ta predmet, vrnemo v
           predstavitvi niza. Uporabnik mora po uporabi sprostiti
           pomnilnik (prostor), v katerem je shranjeno vrnejno
           stevilo. Ce stevilu ne moremo spremeniti predstavitve,
           vrnemo NULL. Vrnjeno stevilo ima obliko
           "<predznak><stevec>/<imenovalec>".
   Parms:  -
   Result: RESULT
   Pre:
   Post:   IF (RESULT <> NULL) THEN RESULT = this ENDIF
*/

// - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
  Cint Stevec();
/* Desc:   Vrne stevec ulomka.
   Parms:
   Result: mStevec
   Pre:
   Post:
*/

// - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
  Cint Imenovalec();
/* Desc:   Vrne imenovalec ulomka.
   Parms:
   Result: mImenovalec
   Pre:
   Post:
*/

// - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
  int Primerjaj(Crac stev);
/* Desc:   Stevilo, ki ga predstavlja ta predmet, primerjamo s stev.
           Vrnemo -1, 0 ali +1 (glej spodaj).
   Parms:  stev -- primerjanec (po vrednosti)
   Result: RESULT
   Pre:
   Post:   IF    (this < stev) THEN RESULT = -1
           ELSIF (this = stev) THEN RESULT =  0
           ELSIF (this > stev) THEN RESULT = +1
           ENDIF
*/

}; // Crac
```

NALOGA: Napišite izvedbeni modul in ga stesirajte.

NAMIG: Uporabite modul `Rac` iz 2. domače naloge.

3. [Za dodatne točke – „bonus naloga"] Razširite kalkulator iz 1. naloge, da bo znal delati z racionalnimi števili. V tem primeru seveda ukaz **D** nima smisla.

Rok za oddajo: *12. 6. 1998. do 17h* v asistentov predal pri vratarju na FMF ali po elektronski pošti.