

NAGYPONTOSSÁGÚ EGÉSZ-ARITMETIKA

TARTALOM

0. A feladat.....	2
1. Az Egész számok ábrázolásai	2
2. A műveletek szignatúrája	3
3. A keretprogram”	4
4. „Technikai” tanácsok.....	7
5. Elegancianövelő lehetőségek a FreePascal-ban – operátorok	7
6. További feladatok	8

0. A FELADAT

... nagypontossággal számolni az (pozitív) egész számok körében. A megvalósítandó műveletek:

- Összeadás
- Kivonás
- Eggyel növelés és csökkentés
- Szorzás
- Maradékos osztás: egész hányados és maradék
- Azonosság reláció
- Rendezési reláció
- Konverziók (String↔nagypontosságú egész)

A fenti műveletek mind **előjel nélküli**, mind **előjeles** változatokra létezzenek.

A mellékelt keretprogram felhasznál egy unitot, amelyben a megoldás részletei található; így maga a program akár késznek is tekinthető lenne. Tisztázzuk az egyes műveletek „alkalmazási szintaktikáját” (=szignatúra), amelyet figyelembe véve kell megírni, és kell elhelyezni a keretprogramban az őket megvalósító függvényeket, ill. eljárásokat. Így minden pillanatban „majdnem” kész programmal végezhető a próbálkozás, és a tetejében tetszőleges sorrendben haladhatunk a fejlesztés során.

A keretprogram „fő szegmense” egy kis (félkész) tesztprogram, amely a kívánt műveleteket „megmozgatja”, csupán az ellenőrzés kedvéért.

Talán hasznos, ha a feladat megoldása előtt rápillantunk egy lehetséges megoldást jelentő [program futására](#).

1. AZ EGÉSZ SZÁMOK ÁBRÁZOLÁSAI

Konstans

```
MaxPont:Egész(255) [=a jegyek maximális száma-1]
```

Típus

```
TSzamJegy=Byte
```

```
TNNEgész=Tömb(0..MaxPont:TSzamJegy)
```

```
TEloJel=(Neg,Poz)
```

```
TEgész=Rekord
```

```
    hossz:0..MaxPont [=a legnagyobb, nem 0 nagyságrend]
```

```
    elojel:TEloJel
```

```
    jegy:TNNEgész
```

```
End
```

2. A MŰVELETEK SZIGNATÚRÁJA

Nem-negatív egészek (**TNNEgesz**) aritmetikája:

Függvény Hosszusag(**Konstans** aNNE:TNNEgesz):Egész
[legnagyobb nem 0 nagyságrend; de aNNE=0 esetén 0]
Eljárás BeolvasNN(**kerd:String; Változó** aNNE:TNNEgesz)
Eljárás KiirNNE(**elo:String; Konstans** aNNE:TNNEgesz;
uto:String)
Függvény EgyenloeNNE(**Konstans** aNNE,bNNE:TNNEgesz):**Logikai**
Függvény NagyobbeNNE(**Konstans** aNNE,bNNE:TNNEgesz):**Logikai**
Eljárás NovelNNE(**Konstans** aNNE:TNNEgesz
Változó bNNE:TNNEgesz)
Eljárás CsokkentNNE(**Konstans** aNNE:TNNEgesz
Változó bNNE:TNNEgesz)
Eljárás OsszeadNNE(**Konstans** aNNE,bNNE:TNNEgesz
Változó cNNE:TNNEgesz)
Eljárás KivonNNE(**Konstans** aNNE,bNNE:TNNEgesz
Változó cNNE:TNNEgesz)
[Ef: aNNE≥bNNE]
Eljárás SzorozNNE(**Konstans** aNNE,bNNE:TNNEgesz
Változó cNNE:TNNEgesz)
[Ef: aNNE≥bNNE]
Eljárás OsztnNNE(**Konstans** aNNE,bNNE:TNNEgesz
Változó hNNE,mNNE:TNNEgesz)
[Ef: bNNE≠0]
Függvény NNE2String(**Konstans** aNNE:TNNEgesz):**String**
[Ef: Hosszúság(aNNE)<256]
Eljárás String2NNE(**Konstans** s:String, **Változó** aNNE:TNNEgesz)
[Ef: 0<Hosszúság(s)<256]

(Előjeles) Egészek (**TEgesz**) aritmetikája:

Eljárás BeolvasE(**kerd:String; Változó** aE:TEgesz)
Eljárás KiirE(**elo:String; Konstans** aE:TEgesz; **uto:String**)
Függvény EgyenloeE(**Konstans** aE,bE:TEgesz):**Logikai**
Eljárás NovelE(**Konstans** aE:TEgesz; **Változó** bE:TEgesz)
Eljárás CsokkentE(**Konstans** aE:TEgesz; **Változó** bE:TEgesz)
Függvény NagyobbeE(**Konstans** aE,bE:TEgesz):**Logikai**
Eljárás OsszeadE(**Konstans** aE,bE:TEgesz; **Változó** cE:TEgesz)
Eljárás KivonE(**Konstans** aE,bE:TEgesz; **Változó** cE:TEgesz)
Eljárás SzorozE(**Konstans** aE,bE:TEgesz; **Változó** cE:TEgesz)
Eljárás OszteE(**Konstans** aE,bE:TEgesz; **Változó** hE,mE:TEgesz)
[Ef: bE≠0]
Eljárás NNE2E(**Konstans** aNNE:TNNEgesz; **Változó** aE:TEgesz)
Függvény SgnE(**Konstans** aE:TEgesz):Egész
Függvény NEHibaE:**Logikai**

A **hibakezelést** (azaz az **NEHiba** unit-beli, nem exportált változó beállítását) most a keretprogramban elvégezni nem lehet, a többit, azaz a lényegi funkciót viszont igen.

Vegye észre a műveletekben alkalmazott „névlogikát”!

3. A KERETPROGRAM

Program NagyEgesz;

```
(*
keretprogram az 1. nagypontosságú gyakorlathoz

Jelölés (a műveletek "postfixumához"):
NNE = Nem negatív egész (azaz nincs előjel)
E   = Egész (előjeles)

Az alábbiakban részletezett eljárások és függvények
(NEHiba Boolean változó, NullaNNE:TPozEgesz konstans)
megtalálhatók az NE_Unit-ban.
A keretprogramot egészítsük ki minél több, a unitban
egyébként definiált alprogrammal (elfedve a unitbeli
definíciójukat)!
```

*)

Uses

```
{$IFDEF FPC -- nem FreePascal=TurboPascal}
Newdelay,
{$ENDIF}
Crt,NE_Unit;

(*
Az NE_Unit interface részének egy részlete:
Function Hosszusag(Const aNNE:TNNEgesz):Integer;
  {legnagyobb nem 0 nagyságrendű azaz -1, ha az értéke 0}
Procedure BeolvasNNE(kerd:String; Var aNNE:TNNEgesz);
Procedure KiirNNE(elo:String; Const aNNE:TNNEgesz; uto:String);
Function EgyenloeNNE(Const aNNE,bNNE:TNNEgesz):Boolean;
Function NagyobbeNNE(Const aNNE,bNNE:TNNEgesz):Boolean;
Procedure NovelNNE(Const aNNE:TNNEgesz; Var bNNE:TNNEgesz);
Procedure CsokkentNNE(Const aNNE:TNNEgesz; Var bNNE:TNNEgesz);
Procedure OsszeadNNE(Const aNNE,bNNE:TNNEgesz; Var cNNE:TNNEgesz);
Procedure KivonNNE(Const aNNE,bNNE:TNNEgesz; Var cNNE:TNNEgesz);
Procedure SzorozNNE(Const aNNE,bNNE:TNNEgesz; Var cNNE:TNNEgesz);
  {Ef: aNNE>=bNNE}
Function NNE2String(Const aNNE:TNNEgesz):String;
  {Ef: Hosszusag(aNNE)<256}
Procedure String2NNE(Const s:String, Var aNNE:TNNEgesz);
  {Ef: 0<Length(s)<256}
{Egész-aritmetika:}
Procedure Beolvase(kerd:String; Var aE:TEgesz);
Procedure Kiire(elo:String; Const aE:TEgesz; uto:String);
Function EgyenloeE(Const aE,bE:TEgesz):Boolean;
Procedure Novele(Const aE:TEgesz; Var bE:TEgesz);
Procedure CsokkentE(Const aE:TEgesz; Var bE:TEgesz);
Function NagyobbeE(Const aE,bE:TEgesz):Boolean;
Procedure Osszeade(Const aE,bE:TEgesz; Var cE:TEgesz);
Procedure Kivone(Const aE,bE:TEgesz; Var cE:TEgesz);
Procedure SzorozE(Const aE,bE:TEgesz; Var cE:TEgesz);
Procedure Oszte(Const aE,bE:TEgesz; Var hanyados,maradek:TEgesz);
  {Ef: bE<>0}

Procedure NNE2E(Const aNNE:TNNEgesz; Var aE:TEgesz);
Function SgnE(Const aE:TEgesz):Integer;
Function NEHibaE:Boolean;

Const NullaNNE:TPozSzam=(0,...,0);
      NullaE:TEgesz=(hossz:0;elovel:Poz;jegy:(0,...,0))
      MaxPont=...; {a jegyek maximális száma-1}
```

```

    Var    NEHiba:Boolean; {volt-e hiba bármilyen nagypontosságú művelet elvégzésekor}
*)

(*
    ide jönnek a fenti rutinok "saját" változatai,
    a fenti szingnatúrával:
*)

(* Pl.:
Function EgyenloenNE(Const aNNE,bNNE:TNNEgesz):Boolean;
Begin
    {...}
End;

Function EgyenloeE(Const aE,bE:TEgesz):Boolean;
Begin
    {...}
End;

Procedure OsszeadNNE(Const aNNE,bNNE:TNNEgesz; Var cNNE:TNNEgesz);
Begin
    {...}
End;

Procedure OsszeadE(Const aE,bE:TEgesz; Var cE:TEgesz);
Begin
    {...}
End;
*)
Const
    IgenNem:Array[Boolean] of String=('nem','igen');
Var
    aNNE,bNNE,cNNE,hNNE,mNNE:TNNEgesz;
    aE,bE,cE,hE,mE:TEgesz;

Begin
    ClrScr;

    {Előjel nélküli egészek tesztje: -----}

    Writeln('---- Előjel nélküli egészek tesztje ----'+UjSor);

    BeolvasNNE('a(NemNegEgész):',aNNE); BeolvasNNE('b(NemNegEgész):',bNNE);

    KiirNNE(UjSor+'',aNNE,''); KiirNNE('=',bNNE,' = ');
    Writeln(EgyenloenNE(aNNE,bNNE));

    KiirNNE('',aNNE,''); KiirNNE('>',bNNE,' = ');
    Writeln(NagyobbenNE(aNNE,bNNE));

    NovelNNE(aNNE,cNNE);
    KiirNNE(UjSor,aNNE,''); KiirNNE('+1 = ',cNNE,'');

    CsokkentNNE(aNNE,cNNE);
If NEHibaE then
Begin
    KiirNNE(UjSor+'Nem lehet csökkenteni ',aNNE,'-t');
End
Else

```

```

Begin
  KiirNNE(UjSor,aNNE,''); KiirNNE('-1 = ',cNNE,'');
End;

OsszeadNNE(aNNE,bNNE,cNNE); KiirNNE(UjSor,aNNE,'');
KiirNNE('+',bNNE,''); KiirNNE(' = ',cNNE,'');

KivonNNE(aNNE,bNNE,cNNE);
If NEHibaE then
Begin
  KiirNNE(UjSor+'Nem lehet kivonni ',aNNE,'-ból/ből '); KiirNNE(' ',bNNE,'-t!');
End
Else
Begin
  KiirNNE(UjSor,aNNE,''); KiirNNE('-',bNNE,''); KiirNNE(' = ',cNNE,'');
End;

SzorozNNE(aNNE,bNNE,cNNE); KiirNNE(UjSor,aNNE,'');
KiirNNE('*',bNNE,''); KiirNNE(' = ',cNNE,'');

OsztNNE(aNNE,bNNE,hNNE,mNNE);
KiirNNE(UjSor,aNNE,''); KiirNNE(' és ',bNNE,'');
Write(' osztása során képződött-e hiba = ',IgenNem[NEHibaE]);
KiirNNE(UjSor,aNNE,''); KiirNNE(' Div ',bNNE,''); KiirNNE(' = ',hNNE,'');
KiirNNE(UjSor,aNNE,''); KiirNNE(' Mod ',bNNE,''); KiirNNE(' = ',mNNE,'');
ReadKey;

{Előjeles egészek tesztje: -----}

Writeln(UjSor+UjSor+'---- Előjeles egészek tesztje ----'+UjSor);

KonvNNE_E(aNNE,aE); KonvNNE_E(bNNE,bE);
KiirE('a(egész):',aE,''); KiirE(' és b(egész):',bE,'');

KiirE(UjSor+UjSor,aE,''); KiirE('=',bE,' = ');
Writeln(EgyenloeE(aE,bE));

KiirE(' ',aE,''); KiirE('>',bE,' = ');
Writeln(NagyobbeE(aE,bE));

NovelE(aE,cE);
KiirE(UjSor,aE,''); KiirE('+1 = ',cE,'');

CsokkenteE(aE,cE);
KiirE(UjSor,aE,''); KiirE('-1 = ',cE,'');

OsszeadE(aE,bE,cE);
KiirE(UjSor,aE,''); KiirE('+',bE,''); KiirE(' = ',cE,'');

KivonE(aE,bE,cE);
KiirE(UjSor,aE,''); KiirE('-',bE,''); KiirE(' = ',cE,'');

OsztE(aE,bE,hE,mE);
KiirE(UjSor,aE,''); KiirE(' és ',bE,'');
Write(' osztása során képződött-e hiba = ',IgenNem[NEHibaE]);
KiirE(UjSor,aE,''); KiirE(' Div ',bE,''); KiirE(' = ',hE,'');
KiirE(UjSor,aE,''); KiirE(' Mod ',bE,''); KiirE(' = ',mE,'');

BeolvasE(UjSor+UjSor+'a(Egész):',aE); BeolvasE('b(Egész):',bE);

KiirE(UjSor+' ',aE,''); KiirE('>',bE,' = ');

```

```

Writeln(NagyobbeE(aE,bE));

SzorozE(aE,bE,cE); KiiE(UjSor,aE,'');
KiiE('*',bE,''); KiiE('= ',cE,UjSor);

Oszte(aE,bE,hE,mE);
KiiE(UjSor,aE,''); KiiE(' és ',bE,'');
Write(' osztása során képződött-e hiba = ',IgenNem[NEHibaE]);
KiiE(UjSor,aE,''); KiiE(' Div ',bE,''); KiiE('= ',hE,'');
KiiE(UjSor,aE,''); KiiE(' Mod ',bE,''); KiiE('= ',mE,'');

ReadKey;
End.

```

4. „TECHNIKAI” TANÁCSOK

A keretprogram letöltése: [keretprogram](#).¹

Szüksége lesz a megoldáshoz az NE_Unit „félíg lefordított kódjaira”.² (Ezek a FP-hez tartoznak.)

Ha mindent (ezt a leírást, a keret forrását, a unit fájljait) egyben meg akar kaparintani, akkor a [tömörített állományt](#) töltsse le!³

5. ELEGANCIANÖVELŐ LEHETŐSÉGEK A FREEPASCAL-BAN – OPERÁTOROK

Itt a „szokásos” kétváltozós, infix jelölésű műveletekről van szó, mint az „azonos-e”, „kisebb-e”, „összeadás” stb. operátorok.

Az ilyen műveleteket ún. **operátorként** –mint egy speciális függvényt– kell definiálni. A definíció *fej sorának* szintaxisa:

```
Operator op (Const x:T1,y:T2) z:T3;
```

Felhasználásának szintaxisa (az „infix”-ség magyarázata):

Var

a:T1;

b:T2;

c:T3;

...

c:=a *op* b;

(Azaz a T1 és a T2 típusú adatra végezzük el az *op* műveletet, amelynek eredménye T3 típusú.)

¹ <http://people.inf.elte.hu/szlavi/InfoRendsz/Nagyarit/Aritmetika/NagyEgesz.pas>

² http://people.inf.elte.hu/szlavi/InfoRendsz/Nagyarit/Aritmetika/ne_unit.ppu és
http://people.inf.elte.hu/szlavi/InfoRendsz/Nagyarit/Aritmetika/ne_unit.o

³ <http://people.inf.elte.hu/szlavi/InfoRendsz/Nagyarit/Aritmetika/NagyPontossagGyak.zip>

Példák:

```
Operator = (Const x, y:TNNE) egyenloE:Boolean;  
Operator > (Const x, y:TNNE) nagyobbE:Boolean;  
Operator + (Const x, y:TNNE) osszeg:TNNE;
```

6. TOVÁBBI FELADATOK

1. Alakítsuk át az eddigi műveleteinket operátor jelölésűvé, és írjuk át a próbaprogramot is ennek szellemében!
2. Keressünk olyan eljárásokat, amelyek –működési módjukat– tekintve inkább függvényként írandók le! (Ilyen például a `String2NNE`.)
3. Egészítsük ki a próbaprogramot a még tesztetlen műveletek megmozgatását célzó részekkel!
4. Gondoljuk meg, milyen szituációk maradtak –esetlegesen– ki a tesztelésből! Tegyük teljessé a tesztelő programot, hogy minél kevesebb inputtal, minél több (lehetőleg az összes) szituáció teszteltté váljon a futtatása révén!