

Nõo Realgümnaasium

Kvalifikatsioonieksam 2004

Koostas: Raivo Laanemets
Juhendaja: Arne Kivimäe

Nõo 2004

Ülesanne

Luu kaks massiivi: arvude_index ja arvud. Esimene neist on indekstabel, mis sisaldab järgmisi välju: nimi (iga arvu tähistatakse tähega), arvu pikkus (byte tüüpi) ning index, mis näitab, kus asub konkreetne arv arvude massiivis. Arvud on esitatud kujul, kus üks bait sisaldab kahte kümnendsüsteemi numbrit. Arvu maksimaalne pikkus on 256 baiti, millest kõrgeim võib sisaldada ka märki määravat osa: negatiivse arvu korral on kõrgeima poolikbaidi sisu 1111_2 , millele vastab 15 kümnendsüsteemis.

Ühe baidi pikkune arv:

0100 0011

| |
4 3

moodustab arvu 43 kümnendsüsteemis

Arvu tähistatakse tabeli indeksis nimega, milleks on üks täht, näiteks

atValjasta('b') väljastab arvu, mille nimi on b, või

atKorruta('a', 'b', 'c') korrutab arvud a ja b ning paigutab tulemuse arvu c.

Korrutamise algoritmi kirjeldus

Arvude korrutamiseks kasutab programm tavalist kirjaliku korrutamise meetodit. See seisneb ühe arvu numbrite pidevas läbikorrutamises teise arvu numbritega, kusjuures nihutatakse iga järgnevat tulemust ühe koha võrra vasakule. Kui kahe numbri korrutise väärtus on suurem kui 9, siis lahutatakse saadud tulemusest 9 ja ülejääk jäetakse meelde ning liidetakse järgmiste numbrite korrutisele. Tulemuse märki sõltub tegurite märkidest: samamärgilised tegurid annavad positiivse korrutise, erimärgilised negatiivse. Korrutamise algoritm on realiseeritud moodulis sredarvd, seda kasutab funktsioon atKorruta, mis asub moodulis arvdtbl.

Ühest baidist saadakse üks kümnendnumber järgmisel viisil:

1) baidi 4 esimest bitti moodustavad soovitud numbri

baidi sisu: 00100111

esimesed neli bitti saadakse kasutades loogilist jaatust (maski)

$00100101 \text{ and } 00001111 = 00000101$,
mis annabki esimese numbri väärtuse baidi kujul

2) baidi 4 viimast bitti moodustavad soovitud numbrit

baidi sisu: 00100111

bitte nihutatakse neli kohta paremale

$00100111 \text{ shr } 4 = 00000010$,
saime numbri väärtuse baidi kujul.

Arvudega tehete tegemise moodul

unit sredarvd;

{

Kvalifikatsioonieksam 2004

Arvudega tehete tegemise funktsioonid.

Funktsioonide nimed algavad eesliitega -sa.

Raivo Laanemets

}

interface

type

TSuurArv=array [0..255] of byte;

{iga kümnendnumber vätab 4 bitti,

255. baidi teine nelik m, r, b m, rgi:

1111- negatiivne arv, mingi teine v, rtus- positiivne}

procedure saSet10Arv(var sis: TSuurArv; osa: word; num: byte); {kümnendarvu
ühe koha seadmine}

procedure saNegArv(var sis: TSuurArv);

procedure saPosArv(var sis: TSuurArv);

procedure saKorruta(sis1, sis2: TSuurArv; var tulem: TSuurArv);

procedure saTyhjendaArv(var sis: TSuurArv);

{arvu m, rgi liigutamine suurima numbri ette}

function saCompress(sis: TSuurArv; var val: TSuurArv): byte;

{arvu m, rgi liigutamine kärkeimale baidile}

procedure saUCompress(sis: TSuurArv; var val: TSuurArv);

function saGet10Arv(sis: TSuurArv; osa: word): byte; {kümnendarvust
ühe osa saamine}

}

function saIsPosArv(sis: TSuurArv): boolean;

function saVordle10(sis1, sis2: TSuurArv): boolean;

procedure saValjasta10Arv(sis: TSuurArv; baidid: byte);

procedure saRndArv(var sis: TSuurArv; len: word);

{kui sis1 on suurem kui sis2, siis

Vordle on true.}

function saLenOf10Arv(sis: TSuurArv): word;

function saBytesOfArv(sis: TSuurArv): byte;

implementation

uses arvdtbl;

```

procedure saUCompress(sis: TSuurArv; var val: TSuurArv);
var
  len: byte;
  tmpnum: byte;
begin
  len:=saBytesOfArv(sis);
  if len>0 then tmpnum:=sis[len-1] else tmpnum:=0;
  val:=sis;
  if tmpnum>240 then {viimane number (baidis) arvus n,,itab m,,rki}
  begin
    if saGet10Arv(sis, (len-1)*2+1)>9 then
    begin
      {writeln('NEG 1');}
      saNegArv(val);
      saSet10Arv(val, len*2-1, 0);
    end;
  end else if (tmpnum and 15)=15 then {eelviimane number (baidis) n,,itab m,,rki}
  begin
    {writeln('NEG 2');}
    val[len-1]:=0;
    saNegArv(val)
  end;
end;

```

```

function saCompress(sis: TSuurArv; var val: TSuurArv): byte;
var
  lastnum: byte;
  len: byte;
begin
  val:=sis;
  if saIsPosArv(sis) then
  begin
    saCompress:=saBytesOfArv(sis);
  end else
  begin
    len:=saBytesOfArv(sis);
    if len>0 then lastnum:=sis[len-1] else lastnum:=0;
    val[255]:=0; {m,,rgi baidi nullimine}
    {writeln('Lastnum:', lastnum);}
    if lastnum>9 then {viimane arv koosneb kahest osast}
    begin

      val[len]:=15; {00001111, kärkeim bait arvus}
      saCompress:=len+1;
    end else {viimane arv koosneb  hest osast}
    begin
      saSet10Arv(val, (len)*2-1, 15);
      saCompress:=len;
    end;
  end;
end;

```

```
    end;  
    end;  
end;
```

```
procedure saRndArv(var sis: TSuurArv; len: word);  
var i: integer;  
begin  
    for i:=0 to len-1 do  
        begin  
            saSet10Arv(sis, i, Random(9));  
        end;  
    end;  
end;
```

```
procedure saValjasta10Arv(sis: TSuurArv; baidid: byte);  
var  
    i: byte;  
    tmp: byte;  
    mask: byte;  
begin  
    if not(saIsPosArv(sis)) then write('-');  
    for i:=baidid downto 0 do  
        begin  
            tmp:=sis[i];  
            write(tmp shr 4);  
            mask:=15;  
            write(tmp and mask);  
        end;  
    end;  
end;
```

```
procedure saTyhjendaArv(var sis: TSuurArv);  
var i: byte;  
begin  
    for i:=0 to 255 do sis[i]:=0;  
end;
```

```
procedure saKorruta(sis1, sis2: TSuurArv; var tulem: TSuurArv);  
var  
    i, j: word;  
    l1, l2: word;  
    neg: boolean; {true- tulemus on negatiivne}  
    arBuf1, arBuf2: byte;  
    accBuf: byte; {arBuf1 ja arBuf2 korrutamisel saadav arv}  
    oldBuf: byte; {nihutamisel vana arvu kontrollimine}  
    yt: byte;  
begin  
    if saIsPosArv(sis1)=saIsPosArv(sis2) then neg:=false else neg:=true;  
    l1:=saLenOf10Arv(sis1); {pikkused hiljem TArv.pikkus j,,rgi}  
    l2:=saLenOf10Arv(sis2);
```

```

{writeln('esimese arvu pikkus:', l1, ' teise arvu pikkus:', l2);}
yt:=0;
saTyhjendaArv(tulem);
{korratamisel tekkiva väimaliku let,,itumise kontrollimine}
if l1+l2>511 then exit
else
  if (sis1[l1-1]+sis2[l2-1]>9) and (l1+l2=510) then
    begin
      { let,,itumise t,,histamiseks seame terve baidi v,,,rtuseks 255}
      tulem[0]:=255;
      exit;
    end;

for i:=0 to l1-1 do
begin
  { yt:=0;}
  arBuf1:=saGet10Arv(sis1, i); {teise arvu numbrid korratatakse selle numbriga}
  for j:=0 to l2 do {maksimaalne let,,itumine, ks koht p,,rast arvu lõpu}
  begin
    arBuf2:=saGet10Arv(sis2, j);
    accBuf:=arBuf1*arBuf2; {tehe}

    {writeln(arBuf1, 'x', arBuf2, '=', accBuf); {readln;}}
    accBuf:=accBuf+yt; {liidame eelmises tulemusel olnud let,,ituja}
    oldBuf:=saGet10Arv(tulem, j+i);
    accBuf:=oldBuf+accBuf; {liidame vana tulemuse}
    {kontrollime, kas toimus let,,itumine}
    saSet10Arv(tulem, j+i, accBuf mod 10); {korratades nihkuvad liidetavad arvud he
värre vasakule}
    yt:=accBuf div 10; { let,,itumise n,,itaja saab uue v,,,rtuse}
  end;
end;
if neg then saNegArv(tulem);
end;

function saBytesOfArv(sis: TSuurArv): byte;
var
  i:integer;
  ok: boolean;
begin
  ok:=false;
  if saGet10Arv(sis, 510)<>0 then
    begin
      {suurim bait sisaldab peale m,,rgi
      ka numbrit}
      saBytesOfArv:=255;
      exit;
    end;
end;

```

```

i:=255;
repeat
  dec(i);
  if sis[i]<>0 then ok:=true;
until ok or (i=0);
if not(ok) then saBytesOfArv:=0 else
  saBytesOfArv:=i+1; {pikkus indeksist 1 v  rra suurem}
end;

```

```

function saLenOf10Arv(sis: TSuurArv): word;
var
  i: word;
  ok: boolean;
begin
  ok:=false;
  i:=511;
  repeat
    dec(i);
    if saGet10Arv(sis, i)<>0 then ok:=true;
  until (i=0) or ok;
  i:=i+1;
  saLenOf10Arv:=i;
end;

```

```

function saVordle10(sis1, sis2: TSuurArv): boolean;
var
  i: word;
  ok: boolean;
  num1, num2: byte;
begin
  saVordle10:=false;
  {1. m,rgi v  rdlus}
  if saIsPosArv(sis1) and not(saIsPosArv(sis2)) then
    begin
      saVordle10:=true;
      exit;
    end;
  {2. pikkuse v  rdlus}
  if saLenOf10Arv(sis1)>saLenOf10Arv(sis2) then
    begin
      saVordle10:=true;
      exit;
    end
  else if saLenOf10Arv(sis1)<saLenOf10Arv(sis2) then
    begin
      saVordle10:=false;
      exit;
    end;
  {3. numbrite v  rdlus}

```



```

i:=saLenOf10Arv(sis1);
ok:=false;
repeat
  dec(i);
  num1:=saGet10Arv(sis1, i);
  num2:=saGet10Arv(sis2, i);
  if num1>num2 then
    begin
      saVordle10:=true;
      exit;
    end;
  if num1<num2 then exit;
until (i=0) or ok;
end;

```

```

procedure saPosArv(var sis: TSuurArv);
begin
  saSet10Arv(sis, 511, 0);
end;

```

```

procedure saNegArv(var sis: TSuurArv);
begin
  saSet10Arv(sis, 511, 15);
end;

```

```

function saIsPosArv(sis: TSuurArv): boolean;
begin
  if saGet10Arv(sis, 511)=15 then saIsPosArv:=false
  else saIsPosArv:=true;
end;

```

```

function saGet10Arv(sis: TSuurArv; osa: word): byte;
var
  nb: byte;
  tmp: byte;
  mask: byte;
begin
  nb:=osa div 2;
  tmp:=sis[nb];
  if (osa mod 2)=0 then {arvuks esimene osa baidist}
  begin
    mask:=15;
    tmp:=tmp and mask;
    saGet10Arv:=tmp;
  end else
  begin
    {tmp:=tmp and mask;}
    tmp:=tmp shr 4;
  end;
end;

```

```
    saGet10Arv:=tmp;  
end;  
end;
```

```
procedure saSet10Arv(var sis: TSuurArv; osa: word; num: byte);
```

```
var
```

```
    nb: byte;  
    tmp: byte;  
    uus: byte;  
    mask: byte;
```

```
begin
```

```
    if osa=65535 then begin writeln('VIGA !! šlet,,itumine saSet10Arv()'); exit; end;
```

```
    nb:=osa div 2; {baidi asukoht sisendis}
```

```
    tmp:=sis[nb];
```

```
    if (osa mod 2) <> 0 then
```

```
    begin
```

```
        mask:=15; {saada esimene osa baidist
```

```
        seatakse teine osa, 00001111
```

```
        arv nihutatakse 4 bitti vaskule}
```

```
        num:=num shl 4;
```

```
    end
```

```
    else mask:=240; {teine osa, 11110000}
```

```
    tmp:=tmp and mask; {nullitakse seatavad bitid}
```

```
    uus:=num +tmp;
```

```
    sis[nb]:=uus;
```

```
end;
```

```
end.
```

Arvude tabeli moodul

```
unit arvdtbl;  
{  
Kvalifikatsioonieksam 2004  
Arvude säilitamiseks mõeldud tabeli kasutamise funktsioonid.  
Funktsioonide nimed algavad eesliitega -at.
```

Raivo Laanemets

```
korrutamine: atKorruta(t1, t2, tulem: char);  
t1 ja t2 on korruatavad arvud ning tulem on  
arvude indekstabelis arvu nimi, kuhu salvestatakse tulemus.  
}
```

```
interface  
uses sredarvd;
```

```
type  
  TArv=record  
    nimi: char;  
    pikkus: byte;  
    index: word;  
  end;
```

```
var  
  arvude_index: array [1..4096] of TArv;  
  arvud: array [1..40960] of byte;  
  arvude_index_i: word; {n,,itab, mitmes on viimane lisatud arv}
```

```
procedure atPuhasta;  
procedure atValjasta(nimi: char);  
procedure atRandom(nimi: char); {juhusliku arvu nimega nimi loomine tabelisse}  
procedure atLisa(sis: TSuurArv; nimi: char);  
procedure atIndexTabel(algus, ridade_arv: word); {arvude indeksi tabeli v,,ljastamine}  
procedure atArvudTabel(algus, arv: word); {arvude tabeli v,,ljastamine}  
procedure atVaataByIndex(var sis: TSuurArv; index: word); {tabelist kohalt index arvu  
võtmine}  
procedure atVaataByNimi(var sis: TSuurArv; nimi: char); {tabelist arvu võtmine arvu nime  
j,,rgi}  
procedure atKorruta(t1, t2, tulem: char); {vt. j,,rgmise kirjeldust}  
function atKorrutaByIndex(indexof1, indexof2: word; nimi: char): word; {kahe arvu  
korrutamine, kui on antud arvude indeksid indekstabelis, annab tulemi  
indeksi, arvu nime ei muudeta (j,,b #0)}
```

```
function atIndexByNimi(nimi: char): word; {arvu indeksi leidmine tema nime j,,rgi,
```

```
kui antud nimega arv puudub tabelist, siis on saadav v,,,rtus 0}
function atLeiaVabaIndex(arvu_pikkus: byte): word; {kohaliku t,,htsusega}
```

implementation

```
procedure atRandom(nimi: char);
var tmp: TSuurArv;
begin
  saTyhjendaArv(tmp);
  saRndArv(tmp, Random(255));
  atLisa(tmp, nimi);
end;
```

```
procedure atValjasta(nimi: char);
var
  arv: TSuurArv;
  i: word;
begin
  atVaataByNimi(arv, nimi);
```

{NB!! koodiläigul puudus m,,rgi toetus 04.2004}

```
  if arv[255]>=240 then write('-');
  for i:=saLenOf10Arv(arv)-1 downto 0 do write(saGet10Arv(arv, i));
end;
```

```
function atIndexByNimi(nimi: char): word;
var i: word;
begin
  atIndexByNimi:=0;
  for i:=1 to arvude_index_i do
    begin
      if arvude_index[i].nimi=nimi then atIndexByNimi:=i;
    end;
  end;
```

```
procedure atKorruta(t1, t2, tulem: char);
begin
  {saaks lisakiirust, kui arvude indeksid leida korruga}
  atKorrutaByIndex(atIndexByNimi(t1), atIndexByNimi(t2), tulem);
end;
```

```
procedure atVaataByNimi(var sis: TSuurArv; nimi: char);
var i: word;
begin
  {indeksi leidmine}
  i:=atIndexByNimi(nimi);
```

```

    if i <> 0 then atVaataByIndex(sis, i);
end;

```

```

function atKorrutaByIndex(indexof1, indexof2: word; nimi: char): word;
var t1, t2, tulem: TSuurArv;
begin
    {arvude saamine tabelist}
    saTyhjendaArv(tulem);
    saTyhjendaArv(t1); saTyhjendaArv(t2);
    atVaataByIndex(t1, indexof1);
    atVaataByIndex(t2, indexof2);
    saKorruta(t1, t2, tulem);
    atLisa(tulem, nimi);
    atKorrutaByIndex:=arvude_index_i; {viimati lisatud Tarvu index}
end;

```

```

procedure atVaataByIndex(var sis: TSuurArv; index: word);
var
    arv: TArv;
    i: word;
    tmp: TSuurArv;
begin
    saTyhjendaArv(tmp);
    saTyhjendaArv(sis);
    arv:=arvude_index[index];
    for i:=arv.index to arv.index +arv.pikkus-1 do {indeks+pikkus+1 reserveeritud m.,rgile}
    begin
        tmp[i-arv.index]:=arvud[i];
    end;
    saUCompress(tmp, sis);
    {sis[255]:=arvud[i+1]; {sis[i-arv.index]:=0;}} {negatiivse arvu probleem 23.04}
end;

```

```

procedure atArvudTabel(algus, arv: word);
var i: integer;
begin
    writeln('-----');
    writeln('Arvud');
    writeln('-----');
    for i:=algus to arv+algus do
    begin
        write(arvud[i], ' ');
    end;
    writeln;
    writeln('-----');
end;

```

```

procedure atIndexTabel(algus, ridade_arv: word);

```

```

var i: integer;
begin
  writeln('----');
  writeln('Arvude indeksid');
  writeln('----');
  for i:=algus to algus+ridade_arv-1 do
  begin
    write('nimi':7); write('pikkus':7); writeln('index':7);
    write(arvude_index[i].nimi: 7);
    write(arvude_index[i].pikkus: 7);
    writeln(arvude_index[i].index: 7);
  end;
  writeln('----');
end;

function atLeiaVabaIndex(arvu_pikkus: byte): word;
var
  i: word;
  index: word;
  pk: word;
begin
  if arvude_index_i=0 then
  begin
    atLeiaVabaIndex:=1;
    exit;
  end else
  begin
    for i:=1 to arvude_index_i do
    begin
      {kontrollime kas antud arvu saab vahele l  kata}
      {pk:=arvude_index[i].pikkus;
      if (pk=0) or (pk>=arvu_pikkus) then }
      index:=arvude_index[i].index+arvude_index[i].pikkus;
    end;
  end;
  atLeiaVabaIndex:=index;
end;

procedure atLisa(sis: TSuurArv; nimi: char);
var
  pk: word;
  uusarv: TArv;
  i: word;
begin
  {indeksi lisamine}
  uusarv.nimi:=nimi;

  saCompress(sis, sis);

```

```

uusarv.pikkus:=saCompress(sis, sis); {!!!!}
uusarv.index:=atLeiaVabaIndex(uusarv.pikkus);
inc(arvude_index_i);

{lisamine arvude massiivi}

for i:=uusarv.index to uusarv.index +uusarv.pikkus do arvud[i]:=sis[i-uusarv.index];
{arvud[i+1]:=sis[255];}

{lisamine arvude indeksite massi}
{inc(uusarv.pikkus);}
arvude_index[arvude_index_i]:=uusarv;
end;

procedure atPuhasta;
var
  i: word;
  arv:TArv;
begin
  arv.nimi:=#0;
  arv.pikkus:=0;
  arv.index:=0;
  for i:=1 to 4096 do arvude_index[i]:=arv;
  for i:=1 to 40960 do arvud[i]:=0;
  arvude_index_i:=0;
end;

end.

```

Testprogramm

```
program kval4;

{
Kvalifikatsioonieksam 2004
demo

Raivo Laanemets
}

{delphis kompileerides tahab APPTYPE direktiivi}
{APPTYPE CONSOLE}

{arvutada on võimalik ka ilma tabelita}
uses sredarvd, arvdtbl;

var
  arv1, arv2: TSuurArv;

begin
  writeln('---PROGRAMMI ALGUS---'); writeln;

  {Arvude tabeli puhastamine (nullide kirjutamine)}
  atPuhasta;

  {arvude arv1 ja arv2 puhastamine}
  saTyhjendaArv(arv1);
  saTyhjendaArv(arv2);

  {arvude 'k,,sitsi' m,,,,ramine k mnendkohtade kaupa}
  saSet10Arv(arv1, 0, 3);
  saSet10Arv(arv1, 1, 1);

  saSet10Arv(arv2, 3, 4);
  saSet10Arv(arv2, 4, 5);

  Randomize;

  {arvu arv1 muutmine negatiivseks}
  saNegArv(arv1);

  {arvu arv1 lisamine tabelisse}
  atLisa(arv1, 'a');

  {teise arvu lisamine tablisse}
```



```
atLisa(arv2, 'b');

{juhusliku arvu c genereerimine tabelisse}
atRandom('c');


{Arvude v, l, jastamine tabelist}
write('Arv a = '); atValjasta('a'); writeln;
write('Arv b = '); atValjasta('b'); writeln;
write('Arv c = '); atValjasta('c'); writeln;


{Tehted}
{a ja b korrutamine, tulemus salvestatakse arvu d}
atKorruta('a', 'b', 'd');
write('Tulemus a x b = '); atValjasta('d'); writeln;


{b ja c korrutamine, tulemus arvu e}
atKorruta('b', 'c', 'e');
write('Tulemus b x c = '); atValjasta('e'); writeln;


writeln;
writeln('---PROGRAMMI LÕPP---');
readln;
end.
```