В этом модуле реализовано несколько процедур длинной арифметики. А именно: сложение, вычитание, умножение и деление. Цель разработки профессионально пригодной библиотеки не ставилась, задача имела чисто учебное назначение, поэтому процедуры реализованы, как обычные операции столбиком. Впрочем, на самом деле их эффективность, как оказалось позже, достаточно хороша и для профессиональных задач. Учебный характер задач выражается еще в ограничениях на процедуры: вычитания, деления и вычисления корня квадратного. В вычитании уменьшаемое должно быть больше вычитаемого, в делении делимое больше делителя, а корень квадратный вычисляется из целого числа и результат также целый.

Задан единый стандарт оформления списка формальных параметров. Первым описывается выводимое данное – результат, за ним идут данные – аргументы. Длинное число представляет собой запись из двух полей: массив цифр и число – количество цифр, начиная с нулевой. Число в массив записывается, начиная с младшего, то есть в обратном порядке в сравнении с общепринятым способом.

Модуль написан на компонентом Паскале.

```
MODULE Arifm;
 IMPORT In, StdLog, Math,Services;
 TYPE
   Numbers=RECORD
     digit:ARRAY 100 OF BYTE;
     len:INTEGER;
   END;
 VAR

PROCEDURE Summ(OUT res:Numbers;IN a,b:Numbers);
 VAR
   i,sum:BYTE;
   max:INTEGER;
BEGIN
  IF a.len>b.len THEN (*Наибольшее a*)
  i:=0;
  sum:=0;
  max:=a.len;
  WHILE i<=max DO
     INC(sum,a.digit[i]);
   IF i<=b.len THEN
     INC(sum,b.digit[i]);
   END;
   IF sum<10 THEN
     res.digit[i]:=sum;
     sum:=0;
   ELSE
      DEC(sum,10);
      res.digit[i]:=sum;
      sum:=1;
    END;
   INC(i);
  END;
  IF sum>0 THEN
    INC(max);
    res.len:=max;
```

```
    res.digit[max]:=sum;
   ELSE
    res.len:=max;
   END;
   ELSE (*Наибольшее b*)
   i:=0;
   sum:=0;
   max:=b.len;
   WHILE i<=max DO
     INC(sum,b.digit[i]);
    IF i<=a.len THEN
     INC(sum,a.digit[i]);
    END;
    IF sum<10 THEN
     res.digit[i]:=sum;
     sum:=0;
    ELSE
      DEC(sum,10);
      res.digit[i]:=sum;
      sum:=1;
    END;
   INC(i);
   END;
   IF sum>0 THEN
    INC(max);
    res.len:=max;
    res.digit[max]:=sum;
   ELSE
    res.len:=max;
   END;
   END;
END Summ;


PROCEDURE Substruction(OUT res:Numbers;IN a,b:Numbers);
VAR
  k,j,i:INTEGER;
BEGIN
  (*Вычитание разрядов b из разрядов a*)
  CopyNumber(res,a);
  i:=0;
  WHILE i<=b.len DO
    IF b.digit[i]<=res.digit[i] THEN
      DEC(res.digit[i],b.digit[i]);
    ELSE
      (*Поиск ненулевого разряда*)
      k:=i+1;
      WHILE res.digit[k]=0 DO
        INC(k);
      END;
      DEC(res.digit[k]);
      j:=i+1;
      WHILE j<k DO
        res.digit[j]:=9;
        INC(j);
      END;
      INC(res.digit[i],10-b.digit[i]);
    END;
    INC(i);
```

```
  END;
  WHILE res.digit[res.len]=0 DO
    DEC(res.len);
  END;
END Substruction;


PROCEDURE Multiply(OUT res:Numbers;IN a,b:Numbers);
VAR
  k,i,c:BYTE;
BEGIN
  (*Инициализация числа - результата*)
  k:=0;
  WHILE k<=a.len+1 DO
    res.digit[k]:=0;
    INC(k);
  END;
  (*Расчет произведения*)
  k:=0;
  WHILE k<=b.len DO
    i:=0;
    WHILE i<=a.len DO
      INC(res.digit[i+k],a.digit[i]*b.digit[k]);
      INC(res.digit[i+k+1],(res.digit[i+k] DIV 10));
      res.digit[i+k]:=SHORT(SHORT(res.digit[i+k] MOD 10));
      INC(i);
    END;
    INC(k);
    res.digit[a.len+k+1]:=0;
  END;
  (*Определение длины произведения*)

  IF res.digit[a.len+b.len+1]#0 THEN
   res.len:=SHORT(SHORT(a.len+b.len+1));
  ELSE
   res.len:=SHORT(SHORT(a.len+b.len));
  END;
END Multiply;

PROCEDURE Division(OUT rem,ost:Numbers; IN a,b:Numbers);
VAR
 L,R,k,c,t,Nz:INTEGER;
 sum:BYTE;
 flag:BOOLEAN;
PROCEDURE proverka():BOOLEAN;
VAR
 k:INTEGER;
BEGIN
  IF ost.digit[R+1]>0 THEN
    RETURN TRUE
  ELSE
  IF L<0 THEN
    RETURN FALSE
  ELSE
   k:=R;
   WHILE k>=L DO
     IF ost.digit[k]>b.digit[b.len+k-R] THEN
        RETURN TRUE;
     END;
```

```
      IF ost.digit[k]<b.digit[b.len+k-R] THEN
        RETURN FALSE;
      END;
      DEC(k);
    END;
    RETURN TRUE;
  END;
 END;
END proverka;
BEGIN
ost:=a;
 IF a.len>=b.len THEN
 ost.digit[ost.len+1]:=0;
 R:=ost.len;
 L:=ost.len-b.len;
 IF ost.digit[R]<b.digit[b.len] THEN (*Сдвиг еще на один разряд*)
   DEC(R);DEC(L);
 rem.len:=a.len-b.len-1;
 ELSE
 rem.len:=a.len-b.len;
 END;
 Nz:=rem.len+1;

IF L>=0 THEN
 WHILE (R>=b.len) DO
   (*Вычисление очередной цифры*)
   sum:=0;
   WHILE proverka() DO
     (*Очередное вычитание*)
     k:=L;
     WHILE k<=R DO

      IF ost.digit[k]>=b.digit[k-L] THEN
        DEC(ost.digit[k],b.digit[k-L]);
      ELSE
       INC(ost.digit[k],10-b.digit[k-L]);
       DEC(ost.digit[k+1]);
      END;
      INC(k);
     END;
     INC(sum);
   END;
   (*Смещение отрезка*)
    DEC(R);
    DEC(L);
    DEC(Nz);
   rem.digit[Nz]:=sum;
END; (*Главный цикл*)
 WHILE (ost.len>0) & (ost.digit[ost.len]=0) DO
  DEC(ost.len);
 WHILE (rem.len>0) & (rem.digit[rem.len]=0) DO
  DEC(rem.len);
 END;
 END;
ELSE
 rem.len:=0;
 rem.digit[0]:=0;
END;
ELSE
```

```
 rem.len:=0;
 rem.digit[0]:=0;
END;
END Division;


PROCEDURE Sqr(OUT res:Numbers;IN arg:Numbers):BOOLEAN;
VAR
  min,max,sr,sm,kv:Numbers;
  k:INTEGER;
BEGIN
  IF (arg.len=0) & (arg.digit[0]=1) THEN
    res.len:=0;res.digit[0]:=1;
    RETURN FALSE;
  END;
  min.len:=0;min.digit[0]:=1;
  Division2(max,arg);
  REPEAT
    Summ(sm,min,max);
    Division2(sr,sm);
    Multiply(kv,sr,sr);
    IF Sr(kv,arg) THEN
      min:=sr;
    ELSE
      max:=sr;
    END;
    Substruction(sm,max,min);
  UNTIL (sm.len=0) & (sm.digit[0]=1);
  Multiply(sm,min,min);
  IF Equal(sm,arg) THEN
   res:=min;
   RETURN FALSE;
  ELSE
   res:=max;
   RETURN TRUE;
  END;
  RETURN FALSE;
END Sqr;

END Arifm.
```