VOLUME 6
2008

SOUTH-WEST
UNIVERSITY
·NEOFIT RILSKY·
BLAGOEVGRAD, BULGARIA

SCIENTIFIC Research

ISSN 1312-7535

ELECTRONIC ISSUE

Some applications of dynamic structures

Mariya Slavcheva Palahanova
South-West University "Neofit Rilski"
Blagoevgrad

Abstract

The way on presenting in the computer memory the large numbers with unlimited precision through linearly dynamic structures is described. The algorithms of the main arithmetical operations are viewed. The passage from the arithmetic of the unlimited precision to the systems for the computer algebra is blocked out.

Basic problems of the numbers with limited precision

By programming mathematical or real practical problems the most often used quantities are real and integer. In the programming languages, each number is represented with fixed number of bytes. For example, the area of the possible whole values depends of the size of the word in the concrete computer. The binary representation of the real numbers is limited by three factors: limited volume of the memory; the area of the real values is nonlinear and uncountable ; the present numbers are allocated irregularly. We save the real numbers in 4, 8 or 10 bytes the most often. When the results from the calculations are very large (or little) numbers, this is related with some disadvantages, because limits the number of the digits.

The only considerations the numbers are being represented in the computer in this way are the speed at work and the present operative memory. The accuracy is a determinant factor in some spheres on the human's cognition but the speed on the calculations is not. In a number of mathematical problems in the realization of algorithms for precisely calculations with numbers and polynomials the standard types of data proves insufficient in the programming languages.

The basic characteristics of the numbers are a record and a meaning. The dependence between the record of the number and his meaning is given in the following description :

1. Whole numbers

Let b be natural number greater from 1. Then each natural number N can in only way be represented in the form $N=a_0*b^0+a_1*b^1+a_2*b^2+\ldots+a_k*b^k$, where $a_0, a_1, a_2,..., a_k \neq 0$ are integers between 0 and b-1. Each natural number N can in only way be written in base-b numeral system $N=\overline{a_k a_{k-1}...a_0}_{(b)}$, where $a_0, a_1, a_2,..., a_k$ are digits by base b. The number N represents themselves by the list (a_0 a_1 a_2... a_k). If the radix b is a power of 10, it can be passed from the common decimal record to a representation by a list quickly and easily. For determination on the sign of the whole number right, additional or inverse code is used. In the right code the whole numbers are identified with the sign "-" in front of the digits of the number. The inverse code and the additional code are using themselves for facilitation of the operations with negative numbers in the personal computers. It existes a short form of a recording on integers in which the digits can be as positives, so as negatives. Each from them consists in the interval $a_i \in [-b/2,b/2)$, where b>2 is the base of the numeral system. The best celebrated system is considered the base-3 symmetric system with base 3 and digits -1, 0, 1. The advantage of such systems is simply performance of operations with negative numbers. They can use themselves for representation on real numbers also.

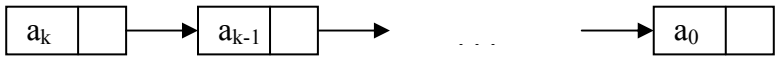
2. Real numbers

Let the real number $(\ldots a_3a_2a_1a_0.a_{-1}a_{-2}a_{-3}\ldots)=\ldots+ a_3*b^3+ a_2*b^2+ a_1*b^1+ a_0*b^0+ a_{-1}*b^{-1}+a_{-2}*b^{-2}+a_{-3}*b^{-3}+\ldots$ is given. Such record of a real number is called a record with a fixing point, e. g. the position of the decimal point is known always. So a record with a floating point

uses themselves for the real numbers, for what the number with precision p(the number of digits in the significant) with a floating point on base b with remainder q is called the pair of numbers (e,f), where (e,f)= f* b^{e-q}. Here e is a whole number (exponent) and f is a fraction number with sign (significant). We suspect, that $|f|<1$, $-b^p< b^p*f< b^p$

In all standard arithmetics in the programming languages the number of the digits of the integers and real numbers is in advance determined. Therefore the numbers represents themselves with a limited accuracy. When the number of the digits is not limited in advance, but it is possible to be limited only from the size of the computer memory, we talk of a unlimited precision. Because the computational memory is a linear structure, and the numbers can be represented with a list, their saving is comfortably in type of related structure- a linear list.

Each structure of data can be presented on an abstract level as a pair <D,R>, where D - an extreme set of elements, even structural data, R - a set from relations, their properties determine the different types of structures of a abstract level. A structure of data is called linear, when the set R consists of one relation “linear arranged”, i.e. each element without her first and her final has only one anterior and one following element. In limiting the number of the operations for access different special cases of a list get themselves: a stack, a queue, a deque, a cyclic queue. They realize themselves in the memory in static type or in dynamic type. The dynamic structures of data change their size during execution of the program and they realize themselves with the help of the mechanism for dynamic distribute of a memory. The number of the elements of the dynamic structure is not defined in advance and can use themselves for representing numbers with unlimited precision. For example, the whole number $N=\overline{a_k a_{k-1}...a_{0(b)}}$ saves itself with the following singly-linked linear list:



Algorithms for the fundamental arithmetical operations over whole numbers with unlimited precision [1]:

Addition and subtraction on two whole numbers $\overline{a_1 a_2...a_{n(b)}}$ and $\overline{c_1 c_2...c_{n(b)}}$. The algorithm for getting of their sum $\overline{d_0 d_1 d_2...d_{n(b)}}$ is the following :

1.j=n и k=0

2.$d_j=(a_j \pm c_j+k) \bmod b$, $k=\lfloor (a_j \pm c_j+k)/b \rfloor$

3. Cycle by an index j. j=j-1. If j>0, return in 2. , otherwise appropriate $d_0=k$ and complete the algorithm.

Multiplication on two whole numbers $\overline{a_1 a_2...a_{n(b)}}$ and $\overline{c_1 c_2...c_{m(b)}}$. The following algorithm is leadding their product $\overline{d_1 d_2...d_{n+m(b)}}$:

1. Appropriate $d_{m+1}=0$, $d_{m+2}=0,\ldots,d_{m+n}=0$ and j=m

2. If $c_j=0$, **then** $d_j=0$ and go in 6.

3. i=n, k=0

4. $t=a_i*c_j+d_{i+j}+k$, $d_{i+j}=t \bmod b$, $k=\lfloor t/b \rfloor$

5. Cycle by an index i. i=i-1. If i>0, then return in 4. , otherwise $d_j=k$.

6. Cycle by an index j. j=j-1. If j>0, then return in 2. , otherwise the implementation of the algorithm carries out itself.

Division on two whole numbers $\overline{a_1 a_2...a_{n+m(b)}}$ and $\overline{c_1 c_2...c_{n(b)}}$. The quotient $\overline{q_0 q_1...q_{m(b)}}$ and the residual $\overline{r_1 r_2...r_{n(b)}}$ are got with the following eight steps:

1. Beginning appropriation $d=\lfloor b/(c_1+1)\rfloor$, $\overline{a_0a_1a_2...a_{n+m(b)}}=\overline{a_1a_2...a_{n+m(b)}}*d$

2. $j=0$

3. Calculating a multiplier s. If $a_j=c_1$, then $s=b-1$, otherwise $s=\lfloor(a_j*b+a_{j+1})/c_1\rfloor$. If the inequality is realized $c_2*s>(a_j*b+a_{j+1}-s*c_1)*b+a_{j+2}$, then $s=s-1$ and we repeat the check-up. We appropriate $q=s$ after the end of all controls.

4. $\overline{a_ja_{j+1}...a_{j+n(b)}}=\overline{a_ja_{j+1}...a_{j+n(b)}}-\overline{c_1c_2...c_{n(b)}}*q$

5. $q_j=q$. If the result on the previous step has been negative, then go to 6. , otherwise go to 7.

6. $q_j=q_j-1$ and $\overline{0c_1c_2...c_{n(b)}}+\overline{a_ja_{j+1}...a_{j+n(b)}}$

7. Cycle by j , $j=j+1$. If $j\le m$, then 3.

8. $\overline{q_0q_1...q_{n(b)}}$ is the searched quotient, the residual is got as a division on $\overline{a_{m+1}...a_{m+n(b)}}$ on d .

Usually these algorithms replay the customary acts at manual addition, subtraction, multiplication and division on whole numbers.

The data structure, singly-linked linear list can be realized by arbitrary programming language. A component of the data describes itself with a structure, the her field consists of own data and pointers with which the elements connect a one with other. The positive integer is described by the algorithmic language Pascal with the following structure:

```
type  pointer= ^ list;
      list = record
               n:integer;
               p:pointer;
             end;
```

In C++ the element of singly-linked list can defined in this way:

```
struct link{
        int n;
        link *p;
        };
```

If the number of components is in advance known can be used a static list – a set from data that arrange themselves in the list consecutively and continuously. In C++ a static list can present with the structure:

```
struct list{
       int k;
       int elem[i];
       };
```

Here k defines the number of the elements in the list, if k=0, then the list is empty, if k=i, then the list is full.

Arrays, recursive structures or recursive classes can be used for the realization of linear lists in C++ . In C++ as object oriented language exist libraries of classes, realizing different structures of data, including and lists. [2]

Linear lists in the systems for computer algebra

If the list (p q) corresponds to the rational number $\frac{p}{q}$, where p and $q\neq 0$ are whole numbers, then the described above way is in effect for the rational numbers. If we supplemente this presentation with corresponding procedures for determining a greatest common divisor of two whole numbers with unlimited precision, for reducing to a common

denominator of two rational numbers, addition and multiplication on rational numbers, then we'll get a rational arithmetic.

The representation with a list can be used for polynomials independently from their type and their power. For example, the polynomial $a_i*x^{i1}+a_2*x^{i2}+\ldots+a_n*x^{in}$, where $i_1>i_2>\ldots>i_n$ are represented with a list from the form $(xa_1i_1a_2i_2\ldots a_ni_n)$. Each coefficient a_i can be a polynomial of another variable or a number with a unlimited precision. The drafted representation for the polynomials can be used for whatever algebraic expressions analogously. [1]

Some translators of Lisp have a built in arithmetic with integer with unlimited precision. Because lists can frame themselves most easily with the specialization language Lisp, for the purpose, a range of systems for computer algebra are written on Lisp or are complemented with the needed resources for a work with lists.

Reference:

[1] Акритас А. ,Основы компьютерной алгебры с приложениями. ,М.: Мир, 1994.
[2] Наков П. , Добриков П. Програмиране=++Алгоритми , София, 2005 .
[3] Абелъсон Х. , Сасман Дж. Структура и интерпретация компьютерных программ , Добросвет 2006
[4] Смит Т.М. , Програмиране на PASCAL - приципи и методи , 1996
[5] Harbour Jonathan S. , Game programming all in one , Boston, MA 02210, http://www.courseptr.com