**Длинная целочисленная арифметика**

В теории чисел могут понадобится вычисления с большими целыми числами. При этом очень важным является сохранение всех значащих цифр. Имеющиеся во всех языках программирования числовые типы имеют ограниченную длину разрядной сетки. При ее превышении происходит либо округление, либо просто отбрасывание последующих цифр.

Реализовать операции с многоразрядными числами без потери значащих цифр можно двумя способами.

а) Запоминать каждое число в массиве. При этом, каждый элемент массива будет хранить одну цифру числа.

б) Запоминать числа в виде строк.

Рассмотрим второй способ.

В этом случае необходимо реализовать базовые алгебраические операции над числами-строками – сложение, вычитание, умножение, целочисленное деление и вычисление остатка от целочисленного деления.

При разработке программ, работающих с длинными числами целесообразно эти базовые операции реализовать в виде отдельных функций и затем уже использовать их в основной программе. Алгоритмы реализации просто повторяют алгоритмы вычислений «столбиком», которые осваиваются в младших классах начальной школы.

Например, для операции сложения:

*В цикле, начиная с последнего символа (цифры) и кончая первым*

*- каждая пара символов преобразуется в числа -* ***х1*** *и* ***х2****;*

*- эти числа складываются*

*- при этом переменная* ***р*** *следит за тем, что не была сумма предыдущей пары чисел больше 9. Если «да», то к х добавляется 1;*

*- если* ***х*** *больше 9, то в качестве результата сложения берется вторая цифра, а переменная* ***р*** *принимает значение “True” (один в уме);*

*- если* ***х*** *меньше 10, переменная р принимает значение “False”;*

*По окончании цикла, если «в уме» еще что-то есть, впереди к результату добавляется 1.*

Если есть возможность выбора между языками программирования, то лучше выбирать Паскаль – уж слишком неуклюжи операции со строками в Бейсике (хотя все это дело вкуса и привычки).

Итак, пусть имеются два длинных числа, представленных в виде строк. Тогда программная реализация операций сложения и вычитания в виде функции может выглядеть следующим образом (Паскаль).

{ Выравнивание длин чисел }

**Procedure Equel(Var s1,s2:string;Var l:integer);**

**Var i,l1,l2:integer;**

**begin**

**l1:=Length(s1);**

**l2:=Length(s2);**

**If l1>l2 then begin**

**l:=l1;**

**For i:=1 to l-l2 do s2:='0'+s2**

**end**

**else begin**

**l:=l2;**

**For i:=1 to l-l1 do s1:='0'+s1**

**end**

**end;**

{ Сложение }

**Function Plus(a,b:string):string;**

**Var i,k,l:integer;**

**x,x1,x2:integer;**

**s,c:string;**

**p:boolean;**

**begin**

**Equel(a,b,l);**

**p:=False;**

**s:='';**

**For i:=l Downto 1 do**

**begin**

**Val(a[i],x1,k);**

**Val(b[i],x2,k);**

**x:=x1+x2;If p then x:=x+1;**

**If x>9 then p:=True else p:=False;**

**x:=x mod 10;**

**Str(x,c);**

**s:=c+s**

**end;**

**If p then s:='1'+s;**

**Plus:=s**

**end;**

{ Вычитание }

**Function Minus(a,b:string):string;**

**Var i,k,l:integer;**

**x,x1,x2:integer;**

**s,c:string;**

**p:boolean;**

**begin**

**Equel(a,b,l);**

**p:=False;**

**s:='';**

**For i:=l Downto 1 do**

**begin**

**Val(a[i],x1,k);**

**Val(b[i],x2,k);**

**If p then x1:=x1-1;**

**If x1<x2 then begin x1:=x1+10;p:=True end**

**else p:=False;**

**x:=x1-x2;**

**Str(x,c);**

**s:=c+s**

**end;**

**While (s[1]='0') and (Length(s)>1) do Delete(s,1,1);**

**Minus:=s**

**end;**

Здесь понадобилась еще одна функция - выравнивание длин складываемых (вычитаемых) чисел. При этом к более короткому числу впереди добавляется нужное количество нулей.

В принципе уже этих функций достаточно для реализации всех остальных алгебраических операций, т. к. операцию умножения можно заменить многократным сложением, а деление - многократным вычитанием.

Например, решить задачу об изобретателе шахмат.

*В качестве вознаграждения за свое изобретение он потребовал на первую клетку положить одно зернышко, на вторую – два, на третью – четыре, на четвертую восемь и т. д. Спрашивается, сколько ему необходимо выдать зерна, если одно зернышко весит 1 г?*

Программная реализация

**Uses Crt;**

**Var a,b:string;**

**i:integer;**

**Procedure Equel …**

**…**

**…**

**Function Plus …**

**…**

**…**

**Begin**

**ClrScr;**

**a:='1';** {Количество зерен в клетке }

**b:=a;** {Общее количество зерен }

**For i:=1 to 63 do**

**begin**

**a:=Plus(a,a);** {Умножение на 2 }

**b:=Plus(b,a)** {Сложение с остальными }

**end;**

**Insert(',',b,Length(b)-5);** {Перевод в тонны – вставка запятой }

**Writeln(b+' тонн');**

**Readkey**

**end.**

Результат – 18446744073709,551615 тонн.

Однако реализация умножения через сложение (деление через вычитание) работает слишком медленно в случае больших сомножителей (или(делимых). Поэтому эти операции все-таки лучше реализовать по школьным алгоритмам. Тем более что они должны использоваться практически во всех заданиях.

Реализация перемножения столбиком (PascalABC).

// Умножение числа на одну цифру

**Function** MultiOne(a:string;b:integer):string;

**Var** i,j:integer;

s,s1:string;

x:integer;

**begin**

s:='';

**For** i:=Length(a) **Downto** 1 **do begin**

x:=StrToInt(a[i]);

x:=x\*b;

s1:=IntToStr(x);

**For** j:=1 **to** (Length(a)-i) **do** s1:=s1+'0';

s:=Plus(s,s1)

**end**;

MultiOne:=s

**end**;

// Перемножение

**Function** Multi(a,b:string):string;

**Var** i,j:integer;

s1,s:string;

**begin**

s:='';

**For** i:=Length(b) **Downto** 1 **do begin**

s1:=MultiOne(a,StrToInt(b[i]));

**For** j:=1 **to** (Length(b)-i) **do** s1:=s1+'0';

s:=Plus(s,s1)

**end**;

Multi:=s

**end**;

**Задания.**

1. Вычислить сумму первых N членов следующего ряда .
2. Составить функцию целочисленного деления длинных чисел.
3. Составить функцию вычисления остатка от целочисленного деления длинных чисел.
4. Составить функцию вычисления целочисленного квадратного корня из длинного числа.
5. Составить функцию вычисления целочисленного корня любой степени из длинного числа.
6. Составить программу вычисления числа сочетаний из N элементов по M.
7. Составить программу вычисления числа размещений из N элементов по M.
8. Вычислить сумму первых N членов следующего ряда .
9. Вычислить сумму первых N членов следующего ряда .
10. Вычислить сумму первых N членов следующего ряда .
11. Вычислить сумму первых N членов следующего ряда .
12. Вычислить сумму первых N членов следующего ряда .
13. Вычислить сумму первых N членов следующего ряда .
14. Вычислить сумму первых N членов следующего ряда .
15. Вычислить сумму первых N членов следующего ряда .