# Лекция 15.
# Моделирование типов (начало)

Стандартные целые типы, существующие в языке, сильно ограничены по диапазону.

Что, если надо посчитать y=50! (факториал). Испытаем существующие целые типы:

## Ограничения

| i! | integer | cardinal | INT64 | UINT64 |
|---|---|---|---|---|
| 1 | 1 | 1 | 1 | 1 |
| 2 | 2 | 2 | 2 | 2 |
| 3 | 6 | 6 | 6 | 6 |
| 4 | 24 | 24 | 24 | 24 |
| 5 | 120 | 120 | 120 | 120 |
| 6 | 720 | 720 | 720 | 720 |
| 7 | 5040 | 5040 | 5040 | 5040 |
| 8 | 40320 | 40320 | 40320 | 40320 |
| 9 | 362880 | 362880 | 362880 | 362880 |
| 10 | 3628800 | 3628800 | 3628800 | 3628800 |
| 11 | 39916800 | 39916800 | 39916800 | 39916800 |
| 12 | 479001600 | 479001600 | 479001600 | 479001600 |
| 13 | 1932053504 | 1932053504 | 6227020800 | 6227020800 |
| 14 | 1278945280 | 1278945280 | 87178291200 | 87178291200 |
| 15 | 2004310016 | 2004310016 | 1307674368000 | 1307674368000 |
| 16 | 2004189184 | 2004189184 | 20922789888000 | 20922789888000 |
| 17 | -288522240 | 4006445056 | 355687428096000 | 355687428096000 |
| 18 | -898433024 | 3396534272 | 6402373705728000 | 6402373705728000 |
| 19 | 109641728 | 109641728 | 121645100408832000 | 121645100408832000 |
| 20 | -2102132736 | 2192834560 | 2432902008176640000 | 2432902008176640000 |
| 21 | -1195114496 | 3099852800 | -4249290049419214848 | 14197454024290336768 |
| 22 | -522715136 | 3772252160 | -1250660718674968576 | 17196083355034583040 |

Оба 4-хбайтовых типа (longint=integer  и longword=cardinal) подходят только для вычисления 12!

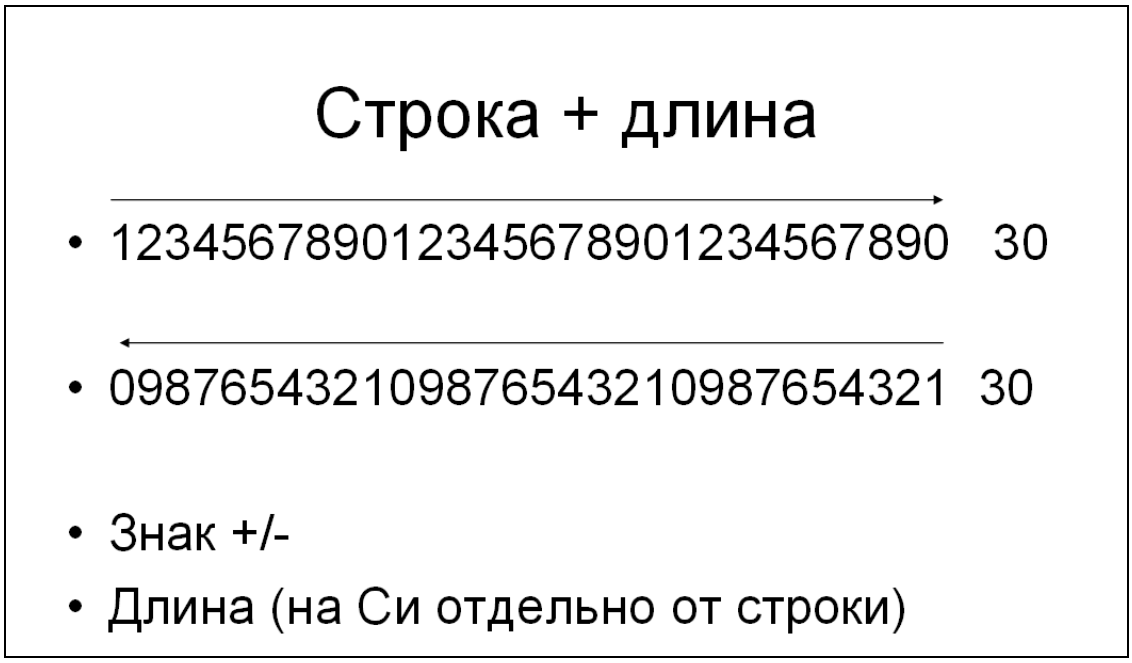
Оба 8-мибайтовых типа (INT64 и UINT64) позволяют без переполнения вычислить 20!, а надо 50!.

Что делать? Создать свой тип: описать структуру и способ хранения, и обязательно определить набор операций для работы с этим типом!

Что надо хранить? Цифры числа, длину, знак.

Причем, удобнее хранить цифры числа в обратном порядке: поскольку все операции производятся с конца числа, то удобнее его сделать началом.

Для хранения числа можно использовать строку из символов-цифр:




Вот как выглядят операции суммирования для чисел, написанных в правильном порядке (слева направо) и наоборот (справа налево):

Также надо выбрать систему счисления

## Система счисления

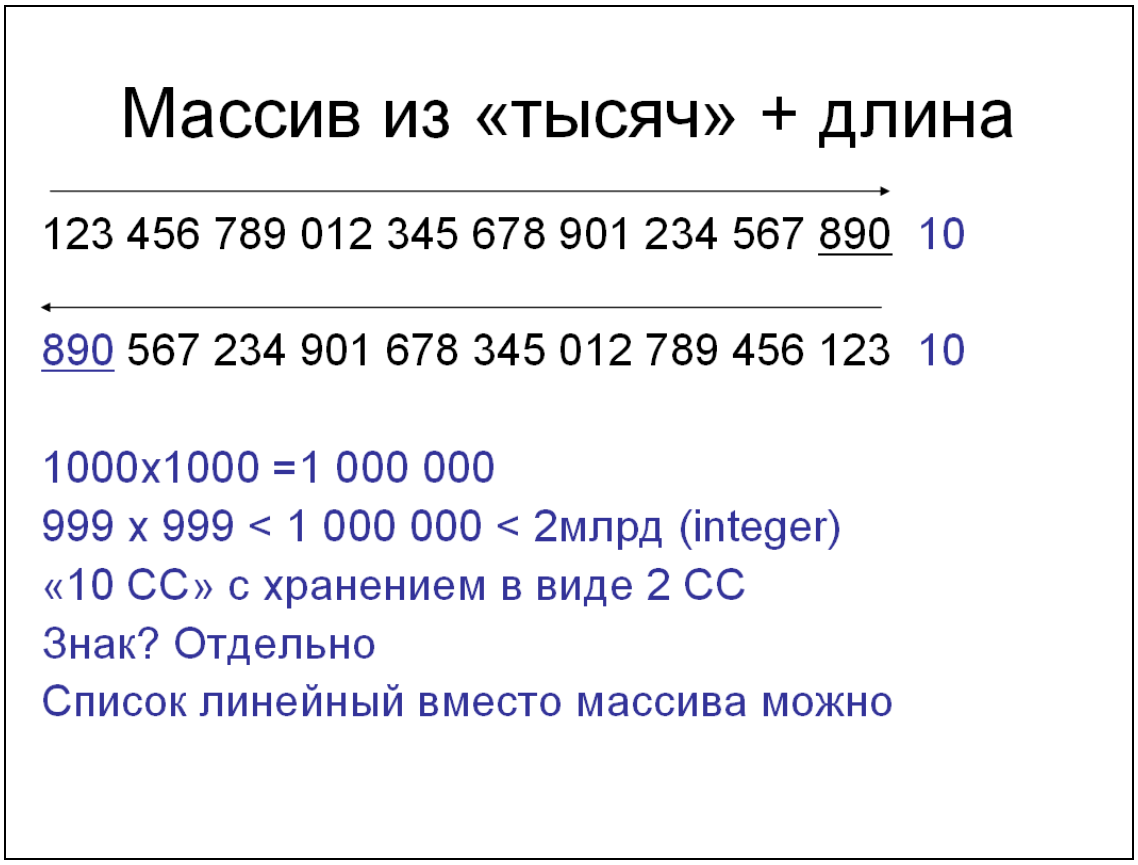
- 10
- 2 (наименьшая таблица умножения, но строка длиннее намного (x4), + перевод СС)  9(10сс)→1001(2сс)
- 16 (сокращенная запись двоичной)
- 32, 64, 128

Умножение в 2СС (двоичной системе счисления)

## Двоичная СС

```
    111111 (2)
X
        101 (2)

---------------
   111111
+
  111111
+
 111111
---------------
100111011
```




Число можно хранить и в массиве. Причем, в каждом элементе удобно хранить не одну цифру числа, а одну триаду – тысячу.

Пример сложения двух чисел, разбитых на триады и расположенные справа налево.

## Массив из «тысяч» + длина

перенос
999 555 333 111  длина=4
+
1  0  0  889  длина=4
---
0  556 333  0  1  длина=5
==========
1 000 333 556 000 при выводе

001 будет храниться в виде 1, но при выводе надо выводить 001, но только если это не первая триада, тогда нули в начале числа не нужны

## Массив из «тысяч» + длина

«нули» при выводе – три знака, если в середине или конце числа
0 1  0  1  длина=4 →
000 001 000 001  длина=4 →
При выводе:
001 000 001 000 →
1 000 001 000

Пример программы с длинным целым на основе короткой строки длиной 70 символов (+0-й). **Сумма двух чисел-строк**:

## Пример. Короткая строка. Delphi

```
program Project1;
{$APPTYPE CONSOLE}
Type Str= String[70];
.....
var s1,s2,s3: Str;

begin
  write('chislo1 = ?'); readln(s1);                                    // строки удобно вводить
  if not Chislo(s1) then begin writeln('не число. ENTER'); readln; exit end;

  write('chislo2 = ?'); readln(s2);
  if not Chislo(s2) then begin writeln('не число. ENTER'); readln; exit end;

  revers(s1); revers(s2);
  s3:=sum(s1,s2); // сумма

  revers(s3);
  writeln('summa  =  ',s3);

  write('Press ENTER'); readln
end.
```

Здесь пропущены процедуры, описанные дальше. Но видно ввод, проверку на отсутствие лишних символов и переворот строки (чтобы цифры в числе шли справа на лево)

## Пример. Короткая строка. Delphi

- Только из цифр – проверка

```
function chislo(var s:Str):boolean;
var i,k: integer;
begin
  Result:=length(s)>0;
  i:=1; k:=length(s);
  while (i<=k) and Result do
  begin
    Result:=s[i] in ['0'..'9'];
    inc(i)
  end;
end;
```

# Пример. Короткая строка. Delphi

- Переворот символов строки

```
procedure revers(VAR s: Str);
var i,k: integer; ch:char;
begin
  k:=length(s);
  for i := 1 to k div 2 do   // значение вычислится 1 раз
  begin
     ch:=s[i];      s[i]:=s[k];      s[k]:=ch;
     dec(k);
  end;
end;
```

# Пример. Короткая строка. Delphi

- Сложение длинной+короткой

```
function sum(s1, s2: Str):Str;
var sl1,sl2: Str; perenos,s, i,k1,k2: integer;
begin
 if length(s2)>length(s1) then
 begin
   sl1:=s2; sl2:=s1
 end
 else
 begin
   sl1:=s1; sl2:=s2
 end;
 k1:=length(sl1); k2:=length(sl2);

Result:='';
 perenos:=0;
 for I := 1 to k2 do
 begin
   s:=Ord(sl1[i])+Ord(sl2[i])+perenos-2*Ord('0');
   perenos:=s div 10;
   s:= s mod 10 + Ord('0');
   Result:=Result+Chr(s);
 end;
 for I := k2+1 to k1 do
 begin
   s:=Ord(sl1[i])+perenos - Ord('0');
   perenos:=s div 10;
   s:= s mod 10 + Ord('0');
   Result:=Result+Chr(s);
 end;

// возможно переполнение
 if perenos>0 then
 begin
   s:= perenos + Ord('0');
   Result:=Result+Chr(s);
 end;
end;
```

```
999
01
-------
90 0 1
```

Числа меняются местами так, чтобы верхнее(первое) было не короче нижнего(второго). Числа складываются числа по длине короткого второго. Затем – перенос складывается с оставшейся частью числа. И в конце – перенос может выйти за пределы самого длинного из чисел, увеличив длину суммы.

Произведение двух чисел

## Пример2. Короткая строка. Delphi

```
program Project1;
{$APPTYPE CONSOLE}
Type Str= String[70];
……..
var s1,s2,s3: Str;

begin
  write('chislo1 = ?'); readln(s1);                                    // строки удобно вводить
  if not Chislo(s1) then begin writeln('не число. ENTER'); readln; exit end;

  write('chislo2 = ?'); readln(s2);
  if not Chislo(s2) then begin writeln('не число. ENTER'); readln; exit end;

  revers(s1); revers(s2);
  s3:=mult(s1,s2); // произведение

  revers(s3);
  writeln('mult    =  ',s3);

  write('Press ENTER'); readln
end.
```

Аналогичный ввод, проверка и переворот чисел до перемножения и после умножения перед выводом результата.

Вспомогательная процедура – умножение числа на одну цифру. Со сдвигом, поскольку эта цифра на самом деле может оказаться сотней или тысячью, или миллионом.

## Пример2. Короткая строка. Delphi

```
function mult1(var sl: Str; const N:integer; const sdvig: integer = 0): Str;
var i,s, perenos: integer;
begin
  Result:='';
  for i := 1 to sdvig do Result:=result+'0';

  perenos:=0;
  for i := 1 to length(sl) do      // возможно переполнение при сдвиге
  begin
    s:=(Ord(sl[i])-Ord('0'))*N+perenos;
    perenos:=s div 10;
    s:= s mod 10 + Ord('0');
    Result:=Result+Chr(s);
  end;

  if perenos>0 then            // возможно переполнение
  begin
    s:= perenos + Ord('0');
    Result:=Result+Chr(s);
  end;
end;
```

И само умножение

## Пример2. Короткая строка. Delphi

```
function mult(s1,s2:Str):Str;
var sl1,sl2: Str; perenos,s, i,k1,k2: integer;
begin
 if length(s2)>length(s1) then
 begin
  sl1:=s2; sl2:=s1
 end
 else
 begin
  sl1:=s1; sl2:=s2
 end;
 k1:=length(sl1); k2:=length(sl2);

 Result:='0';
 for i := 1 to k2 do
  Result:=sum(Result, mult1(sl1, Ord(sl2[i])-Ord('0'), i-1));
end;
```

Теперь можно искать 50!

Попробуем сумму и произведение:


А вот и факториал (до 53!, дальше – опять переполнение на 70-символьном числе):