**Анализ на решението на задача
D3. Израз**

Опит да изчислим израза с използване на функцията *pow*(2, *n*) ще даде грешен резултат за големи стойности на *n*. Налага се да напишем някакъв вариант на аритметика с дълги числа. От израза за изчисление е видно, че трябва да се реализират действията умножение (с чиято помощ ще извършим степенуването в израза), събиране и изваждане. Всяка цифра на резултата ще съхраняваме в отделен елемент на масив с цели числа. За да определим размера на масива, нека разгледаме следната таблица с резултати на 2*n*

21 = 2

22 = 4

23 = 8

24 = 16

25 = 32

26 = 64

27 = 128

...

Забелязваме, че резултатът първоначално е едноцифрен. При увеличаване на *n* с 3, цифрите на резултата се увеличават с 1. Следователно броят на цифрите в резултата за n = 10000 е приблизително 10000:3. Използваме това число като начално приближение за размера на масива. В последствие ще се окаже, че цифрите на 210000 са дори по-малко (3011). В граничния случай, когато $m=-2^{n}$, трябва на практика да удвоим резултата, което потенциално може да изиска още една допълнителна цифра – дефинираме масива за резултат с 3012 елемента. Съобразяваме, че *m* може да е дълго число и затова трябва да го прочетем като низ – необходими са още два допълнителни знака в низа за знака на числото и за „терминиращата“ нула.

**Степенуването (многократното умножение с 2)**

Сега да реализираме действието умножение на дълги числа с 2, с чиято помощ ще извършим степенуването в израза. Първоначално в резултата записваме 2, т.е. 21. Остава да умножим резултата *n*-1 пъти с 2. Умножаването на резултата с 2 правим като всяка от цифрите му умножаваме с 2 и добавяме евентуалния пренос, получен от операцията с предишната (по-младша) цифра. Ако полученото е двуцифрено число, формираме пренос за следващата (по-старша) цифра в резултата, а като текуща цифра записваме само дясната цифра от полученото. След извършване на умножението с 2 на най-старшата (най-лявата) цифра на резултата е възможно да се формира пренос – разширяваме резултата с една цифра и в нея записваме преноса.

След извършване на степенуването трябва да извадим от резултата *m*. Съобразяваме, че когато *m* е отрицателно, $2^{n}-m$ е еквивалентно на $2^{n}+\left|m\right|$, откъдето следва, че трябва да реализираме действията събиране и изваждане с дълги числа.

**Събирането**

Всяка цифра на резултата събираме с цифрата на *m* в същата позиция като отчитаме и евентуален пренос от операцията в съседната по-младша позиция. Ако полученото е двуцифрено число, формираме пренос за следващата (по-старша) цифра в резултата, а като текуща цифра записваме само дясната цифра от полученото. След извършване на събирането с 2 на най-старшата (най-лявата) цифра на *m* е възможно да се формира пренос към по-старша цифра в резултата. Ако има такава цифра, добавяме преноса към нея, което може да доведе до възникване на нов пренос към следващата по-старша цифра – каскадно обработваме преноса. Ако възникне пренос и при най-старшата позиция в резултата (най-лявата) – разширяваме резултата с една цифра и в нея записваме преноса.

**Изваждането**

Запопчвайки от дясно наляво извършваме действието изваждане на цифрата на *m* в текущата позиция от цифрата на резултата в същата позициия, като отчитаме и евентуално наличие на заем (намаляваме в такъв случай предварително цифрата на резултата с 1). Ако полученото е отрицателно, формираме заем от следващата позиция, а цифрата на резултата в текущата позиция увеличаваме с 10. След обработване на всички цифри на *m* е възможно да остане необработен заем – обработваме го каскадно, подобно на начина при събирането. Съобразяваме, че след изваждането в резултата може да са възникнали водещи нули – елиминираме ги, като намаляваме логическия размер на масива.

Сорс-кодът на програмата

#include <iostream>

#include <cstring>

**using namespace std**;

**char** A[3012] = {2}, **/// Резултатът (първоначално: 2, т.е. 2\*\*1)**

 M[3013];

**int** len\_A=1, **/// Брой цифри в резултата**

 len\_M, **/// Брой цифри в M**

 n, i, j,

 cf, **/// Пренос/заем към/от по-старши разряд при аритм. опер.**

 nf; **/// M е положително (при nf!=0 е отрицателно)**

**int** main () {

 **cin** >> n >> M; **/// Входните данни**

 **/// Степенуването в израза (n-1 пъти умножава резултата с 2)**

 **while** (--n) { **/// n-1 пъти умножаваме резултата с 2.**

 **for** (i=cf=0; i<len\_A; i++){ **/// Самото умножение на цифрите на резултата:**

 A[i] = 2 \* A[i] + cf; **/// Умножава поредната цифра с 2 с отчитане на преноса.**

 cf = A[i] / 10; **/// Формира пренос към следващата (по-старша) цифра,**

 A[i] %= 10; **/// а резултатът в позицията остава едноцифрен.**

 }

 **if** (cf) A[len\_A++] = cf; **/// ако след умножението на най-старшата цифра с 2**

 **/// възниква пренос - разширяваме резултата с една**

 } **/// цифра и в нея записваме преноса**

 **/// Подготвя M за извъшване на следващите действия в израза**

 len\_M = strlen(M); **/// Брои знаците (разрядите) в M**

 **for** (i=0, j=len\_M-1; i<j ; i++,j--) **/// Обръща низа M**

 **swap**(M[i],M[j]);

 nf = M[len\_M-1]=='-'; **/// Вдига флага nf, ако M е отрицателно,**

 **if** (nf) len\_M--; **/// и скъсява М само до цифровите символи**

 **/// Изважда М от 2\*\*n**

 **if** (nf) { **/// Изважда отрицателно число (т.е. събира с |M|)**

 **for** (i=cf=0; i<len\_M; i++){

 A[i] += M[i] - '0' + cf; **/// Добавя цифрата на M към съответната цифра на резултата**

 **/// с отчитане на преноса. Деиствието (M[i] - '0') е
 /// необходимо защото цифрите на М са в "текстов" формат**

 cf = A[i] / 10; **/// Формира евентуален пренос към по-старша цифра,**

 A[i] %= 10; **/// а резултатът в позицията остава едноцифрен.**

 }

 **for** ( ; cf; i++){ **/// Aко е останал пренос след събирането:**

 **if** (i==len\_A) len\_A++; **/// Разширява резултата с 1 цифра (при необходимост)**

 A[i] += cf; **/// и добавя преноса към цифрата на резултата.**

 cf = A[i] / 10; **/// Формира пренос към следващата (по-старша) цифра,**

 A[i] %= 10; **/// а резултатът в позицията остава едноцифрен.**

 }

 }

 **else** { **/// Изважда положително число**

 **for** (i=cf=0; i<len\_M; i++){

 A[i] -= M[i] - '0' + cf; **/// Изважда цифрата на M от съответната цифра на резултата**

 **/// с отчитане на заема извършен от по-младша цифра.**

 cf = A[i] < 0; **/// Формира евентуален заем от по-старша цифра**

 **if** (cf) A[i] += 10; **/// и преобразува текущата цифра в "положителна"**

 }

 **for** ( ; cf; j++){ **/// Aко е останал заем след изваждането,**

 A[i] -= cf; **/// изважда го от поредната цифра на резултата,**

 cf = A[i] < 0; **/// формира евентуален заем от по-старша цифра и**

 **if** (cf) A[i] += 10; **/// и преобразува текущата цифра в "положителна"**

 }

 **while** (len\_A>1 and A[len\_A-1]==0) len\_A--;**/// маха "водещите" нули след изваждането**

 }

 **for** (i=len\_A; i; **cout**<<(**int**)A[--i]); **cout** << **endl**; **/// Извежда резултата**

 **return** 0;

}

*Автор: Евгений Василев*