**P1001**

当两个比较大的整数相乘时，可能会出现数据溢出的情形。为避免溢出，可以采用字符串的方法来实现两个大数之间的乘法。具体来说，首先以字符串的形式输入两个整数，每个整数的长度不会超过8位，然后把它们相乘的结果存储在另一个字符串当中（长度不会超过16位），最后把这个字符串打印出来。例如，假设用户输入为：62773417和12345678，则输出结果为：774980393241726.  
  
输入：  
　　62773417 12345678  
  
输出：  
　　774980393241726

C++代码：

# include <stdio.h>

int main()

{

long long a,b;

scanf("%lld%lld", &a, &b);

printf("%lld", a\*b);

return 0;

}

C代码：

/\*

问题描述

　　在C/C++语言中，整型所能表示的范围一般为-231到231（大约21亿）,即使long long型，一般也只能表示到-263到263。要想计算更加规模的数，就要用软件来扩展了，比如用数组或字符串来模拟更多规模的数及共运算。

　　现在输入两个整数，请输出它们的乘积。

输入格式

　　两行，每行一个正整数，每个整数不超过10000位

输出格式

　　一行，两个整数的乘积。

样例输入

99

101

样例输出

9999

数据规模和约定

　　每个整数不超过10000位

\*/

#include<stdio.h>

#include<string.h>

#define V\_LEN ( 20000 ) // 结果长度

void chengfa/\*乘法\*/( char \*a, char \*b, int \*value );

void fanzhi/\*反置对齐\*/( char \*str );

void swap( char \*a, char \*b );

void jw/\*进位\*/( int \*str );

void shuchu/\*输出结果\*/( int \*value );

int main(void){

char a[ 10001 ], b[ 10001 ];

int value[ V\_LEN ] = { 0 };

scanf("%s%s", a, b);

// gets( a ); // 此种写法在蓝桥网的训练环境下会出错

// gets( b );

chengfa/\*乘法\*/( a, b, value );

shuchu/\*输出结果\*/( value );

return 0;

}

void shuchu/\*输出结果\*/( int \*value ){

int \*p = value + V\_LEN - 1; // p指向最高位

while( \*p == 0 ){

p --;

}

if( p < value ){ // 越界

putchar( '0' );

} else {

while( p >= value ){

printf("%d", \*p--);

}

}

putchar('\n');

}

void jw/\*进位\*/( int \*value ){

int i;

for ( i = 0 ; i < V\_LEN ; i ++ ) {

value[ i + 1 ] += value[ i ] / 10;

value[ i ] %= 10;

}

}

void swap( char \*a, char \*b ){

char t = \*a;

\*a = \*b;

\*b = t;

}

void fanzhi/\*反置对齐\*/( char \*str ){

int len = strlen( str );

int i;

for ( i = 0 ; i < len / 2 ; i ++ ) {

swap( str + i, str + len - 1 - i );

}

}

void chengfa/\*乘法\*/( char \*a, char \*b, int \*value ){

fanzhi/\*反置对齐\*/( a );

fanzhi/\*反置对齐\*/( b );

int len1 = strlen( a ), len2 = strlen( b );

int i;

for ( i = 0 ; i < len2 ; i ++ ) { // 遍历表达式 2

int j;

for ( j = 0 ; j < len1 ; j ++ ) { // 遍历表达式 1

value[ i + j ] += ( b[ i ] - '0' ) \* ( a[ j ] - '0' ); // 加值

}

}

jw/\*进位\*/( value );

}

JAVA代码：

import java.io.\*;

import java.util.\*;

import java.math.\*;

import java.text.\*;

public class Main {

void solve() {

BigInteger a, b;

a = new BigInteger(in.next());

b = new BigInteger(in.next());

out.println(a.multiply(b));

}

FastScanner in;

PrintWriter out;

void run(String s) {

if (s == null) {

in = new FastScanner();

} else {

in = new FastScanner(s);

}

out = new PrintWriter(System.out);

solve();

out.close();

}

public static void main(String[] args) {

new Main().run(null);

}

static class FastScanner {

BufferedReader br;

StringTokenizer st;

FastScanner(String file) {

try {

br = new BufferedReader(new FileReader(file));

} catch (FileNotFoundException e) {

e.printStackTrace();

}

}

FastScanner() {

br = new BufferedReader(new InputStreamReader(System.in));

}

public String next() {

while (st == null || !st.hasMoreTokens()) {

try {

st = new StringTokenizer(br.readLine());

} catch (IOException e) {

e.printStackTrace();

}

}

return st.nextToken();

}

boolean hasMoreTokens() {

while (st == null || !st.hasMoreTokens()) {

try {

String s = br.readLine();

if (s == null)

return false;

st = new StringTokenizer(s);

} catch (IOException e) {

e.printStackTrace();

}

}

return true;

}

public int nextInt() {

return Integer.parseInt(next());

}

public long nextLong() {

return Long.parseLong(next());

}

public double nextDouble() {

return Double.parseDouble(next());

}

}

}