Assignment One 说明文档

1 分析

- ***1.1 问题重述***

  分别输入 n 和 k，输出应满足如下要求：

  1. 每一个数字都是n位数，且为非负整数
  2. 相邻两数位上的值的差的绝对值等于k
  3. 每个数字不能带有前导零（例：01）
  4. 有多个数字满足要求可输出时，按升序排列，用逗号分隔

- ***1.2 参数分析***

  $n \geq 2$

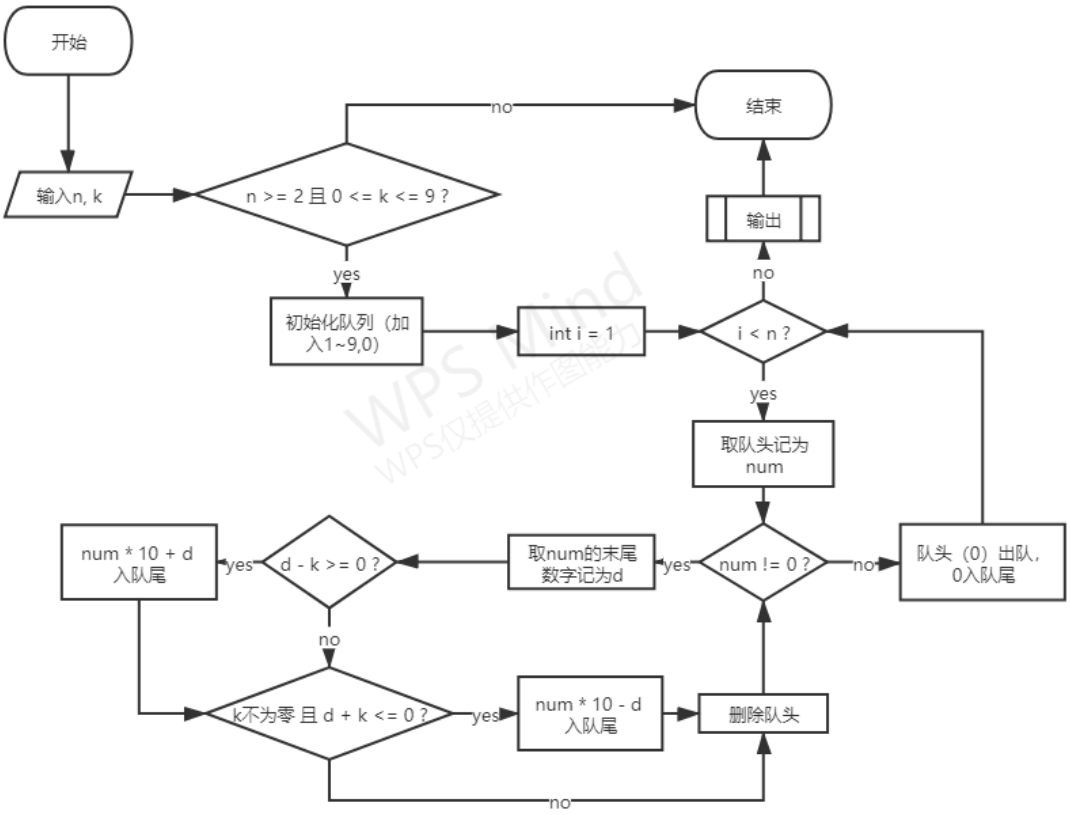
  $0 \leq k \leq 9$

2 算法设计

- **2.1 思路**

  - 采用BFS（广度优先），从首位开始，每层试着在末尾加一位数字，一个数一个数地构造符合要求的答案
  - 由于不能有前导零，第一位数字最多有 9 种选择 $\{1,2,3,4,5,6,7,8,9\}$；其他位的数字最多有 2 种选择（加k / 减k）。这意味着 n 位数最多有 $9 \times 2^{n-1}$ 种情况
  - 一个 n 位的数字可以看作 $n-1$ 位数加上最后一位数字，如果 $n-1$ 位数以数字 d 结尾，则 n 位数字将以 $d-k$ 或 $d+k$ 结尾（前提：在 $[0,9]$ 范围内）

- ## 2.2 流程图



- ## 2.3 实现

  - ### 细节说明：

    - 做**合法性判断**，只有参数合法才能进行以下过程，否则输出为空
    - 采用了数据结构**队列**，初始时依次加入**可能的**首位数字 $1,2,3,4,5,6,7,8,9$
    - 向队列插入**0**，由于0不可能成为合理的输出，在此利用这个特性将其作为每层循环的**结束符**（结束标志）
    - 利用队列先进先出（**FIFO**）的特点，每出一个队头（i 位数），则检查该队头（不为0时）能否拓展为符合要求的 $(i+1)$ 位数，若能则将拓展后的 $(i+1)$ 位数插入队尾；

      拓展方法：

      1. 记： num（当前的 i位数） d（当前的 i位数的末尾数字）；
      2. $d-k$ 或 $d+k$ 在$[0,9]$范围内， 则 $num\times 10+d-k$ 或 $num\times 10+d+k$ 就是要找的 $(i+1)$ 位数；
      3. 按升序原则，先检查$d-k$，再检查$d+k$；
      4. 在检查 $d+k$ 时，应同时检测 k 是否等于 0，因为若 k 等于 0而不进行该层判断，将会在队列中插入重复的数。
    - 当取出的队头为 0 时，表示这一层循环结束，往队尾插入 0 作为**下一层循环**的结束符，进入下一层循环
    - 如此，第 $i(1\le i\le (n-1))$ 层循环的过程，实质上是队列元素从从 i 位数到 $(i+1)$ 位数的**拓展**过程；

      第 i层循环结束后（第 $(i+1)$层循环开始前），队列中的元素都是 $(i+1)$位数（在这里不考虑结束符 0）；

$(n-1)$层循环后，队列中的元素便是符合要求的 n位数了（除去队尾元素 0），进行输出

- **输出**时，依次输出队头元素，直到队头为 0（0不输出）；最后一个数字输出时，后面不接逗号

主函数代码实现：

```
int main()
{
    //非负整数有n位，任意相邻两位上的数字之差（绝对值）为k
    int n, k;
    cout << "请输入n，k：" << endl;
    cin >> n >> k;

    if (n >= 2 && k >= 0 && k <= 9) {    //合法性判断
        queue<int> que;
        for (int i = 1; i <= 9; i++) {  //初始化队列，先加入（可能的）首位数字
            que.push(i);
        }
        que.push(0);                        //0不可能是输出，将其作为每次循环的结束符

        //构建
        int num, d;                         //当前数字以及其末尾数字
        for (int i = 1; i < n; i++) {    //循环n-1次
            while ((num = que.front()) != 0) {
                d = num % 10;
                //按大小顺序依次加入
                if (d - k >= 0) {
                    que.push(num * 10 + d - k);
                }
                if (k != 0 && d + k <= 9) {
                    que.push(num * 10 + d + k);
                }
                que.pop();
            }
            que.pop();          //队头的0出队
            que.push(0);        //0入队尾
        }

        //输出
        if (que.front() != 0) {
            cout << que.front();
            que.pop();
        }
        while (que.front() != 0) {
            cout << "," << que.front();
            que.pop();
        }
    }

    return 0;
}
```

- **2.4 复杂度分析**

时间复杂度	空间复杂度
$O(2^n)$	$O(2^n)$

---

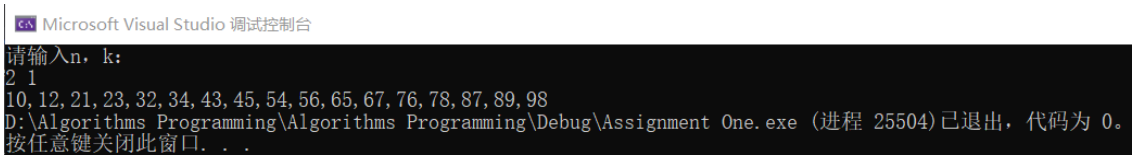
3 测试

- ***3.1 功能测试***

  case 1:

  ```
  Input: n = 2, k = 1
  Output: [10,12,21,23,32,34,43,45,54,56,65,67,76,78,87,89,98]
  ```

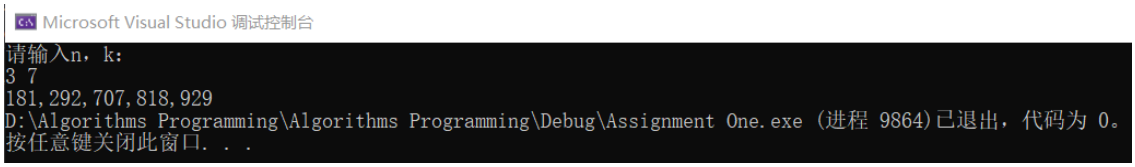
  运行结果：

  case 2:

  ```
  Input: n = 3, k = 7
  Output: [181,292,707,818,929]
  ```

  运行结果：

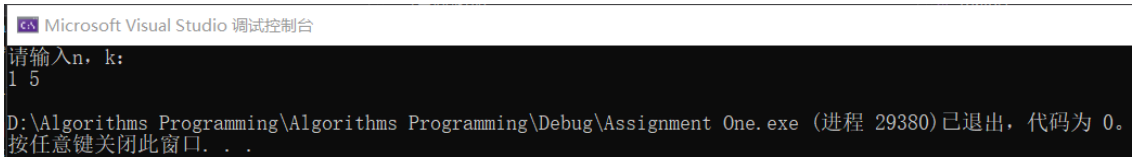
  

- ***3.2 边界测试***

  case 1:

  ```
  Input: n = 1, k = (any value)
  Output: []
  ```

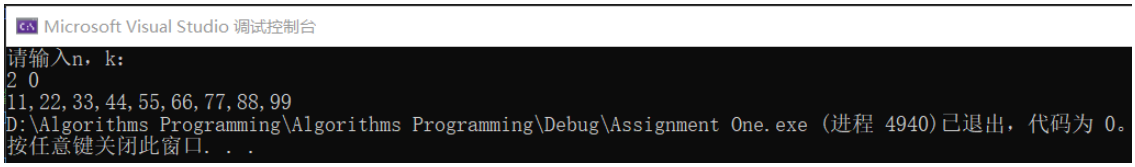
  运行结果：



case 2:

```
Input: n = 2, k = 0
Output: [11,22,33,44,55,66,77,88,99]
```

运行结果：



- ## *3.3 性能测试*

将主程序插入在如下基准程序的指定位置，来测试时间性能。

测试环境： PC机（Legion R7000 2020）

处理器（AMD Ryzen 5 4600H with Radeon Graphics 3.00 GHz）

机带RAM（16.00 GB）

操作系统（Windows 10 家庭中文版 64位）

IDE（Visual Studio 2019 社区版）

```
#include <iostream>
#include <queue>
#include <iomanip>
#include <cstdio>
#include <windows.h> //取系统时间
using namespace std;

int main()
{
    LARGE_INTEGER tick, begin, end;

    QueryPerformanceFrequency(&tick);    //获得计数器频率
    QueryPerformanceCounter(&begin);     //获得初始硬件计数器计数

    /* 此处是主程序开始 */


    /* 此处是主程序结束 */

    QueryPerformanceCounter(&end);       //获得终止硬件计数器计数

    cout<<endl;
    cout << "计数器频率 : " << tick.QuadPart << "Hz" << endl;
    cout << "计数器计数 : " << end.QuadPart - begin.QuadPart << endl;
```

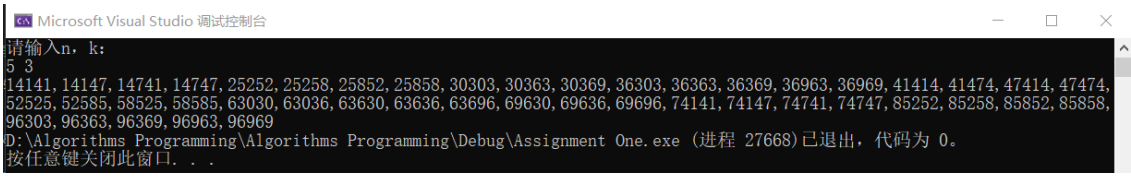
```
    cout << setiosflags(ios::fixed) << setprecision(6) <<
double(end.QuadPart - begin.QuadPart) / tick.QuadPart << "秒" << endl;

    return 0;
}
```

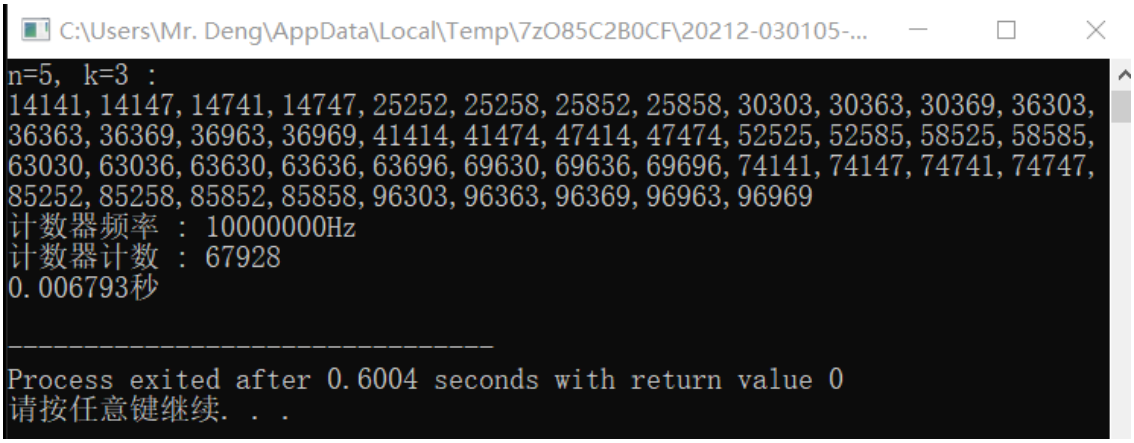
case 1:

```
Input: n = 5, k = 3
Output: (参见如下程序运行截图)
```

运行结果:



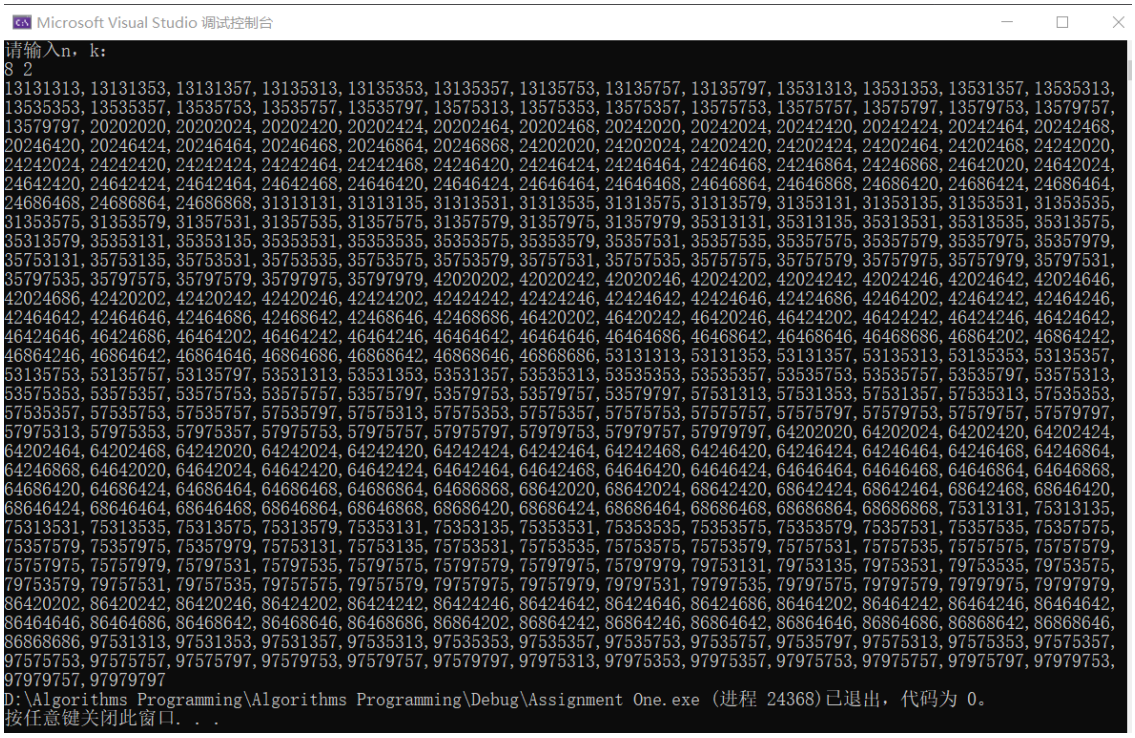
运行时间：$6.793ms$



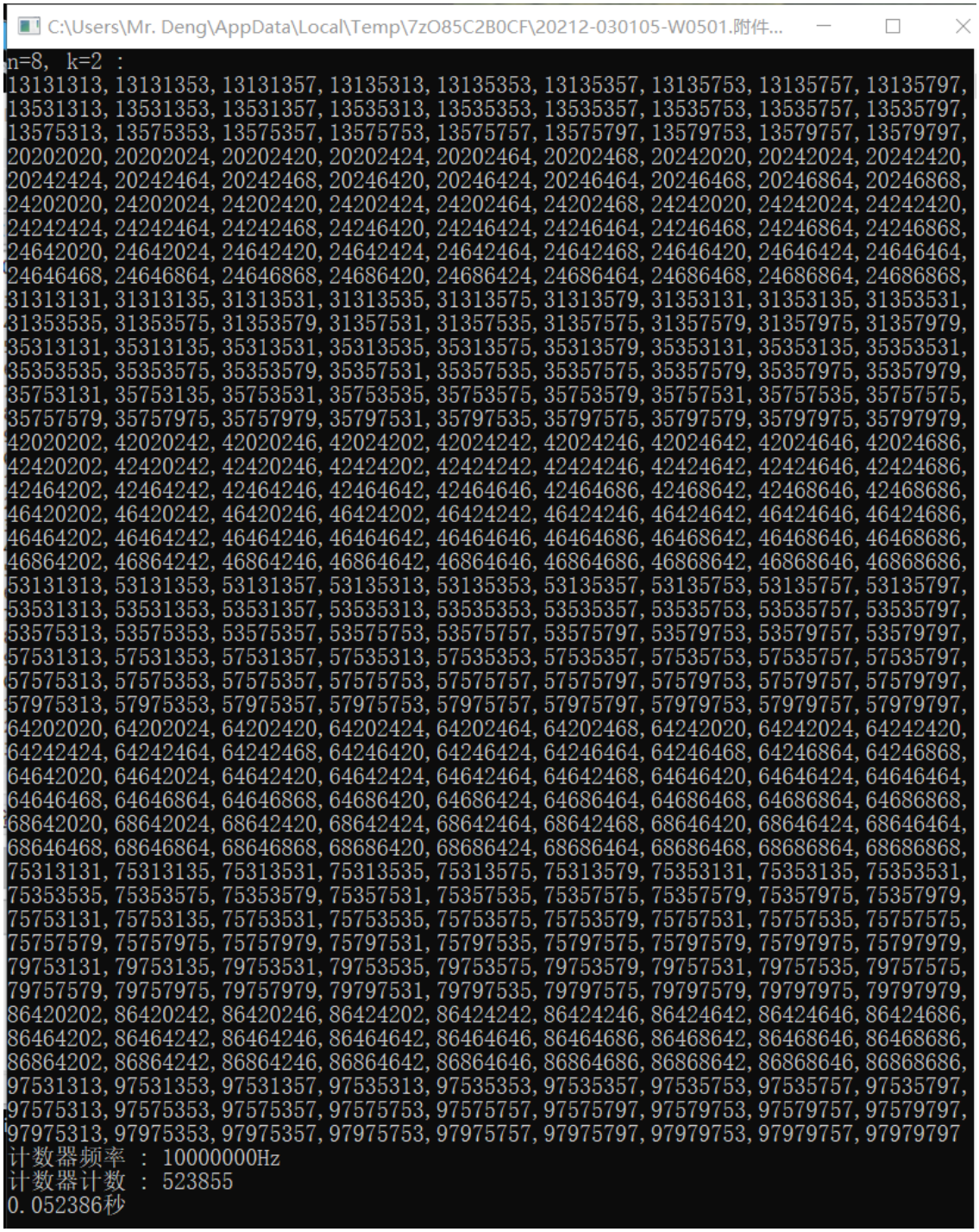
case 2:

```
Input: n = 8, k = 2
Output: (参见如下程序运行截图)
```

运行结果:



运行时间：$52.386ms$


C:\Users\Mr. Deng\AppData\Local\Temp\7zO85C2B0CF\20212-030105-W0501.附件...
n=8, k=2 :
13131313, 13131353, 13131357, 13135313, 13135353, 13135357, 13135753, 13135757, 13135797,
13531313, 13531353, 13531357, 13535313, 13535353, 13535357, 13535753, 13535757, 13535797,
13575313, 13575353, 13575357, 13575753, 13575757, 13575797, 13579753, 13579757, 13579797,
20202020, 20202024, 20202420, 20202424, 20202464, 20202468, 20242020, 20242024, 20242420,
20242424, 20242464, 20242468, 20246420, 20246424, 20246464, 20246468, 20246864, 20246868,
24202020, 24202024, 24202420, 24202424, 24202464, 24202468, 24242020, 24242024, 24242420,
24242424, 24242464, 24242468, 24246420, 24246424, 24246464, 24246468, 24246864, 24246868,
24642020, 24642024, 24642420, 24642424, 24642464, 24642468, 24646420, 24646424, 24646464,
24646468, 24646864, 24646868, 24686420, 24686424, 24686464, 24686468, 24686864, 24686868,
31313131, 31313135, 31313531, 31313535, 31313575, 31313579, 31353131, 31353135, 31353531,
31353535, 31353575, 31353579, 31357531, 31357535, 31357575, 31357579, 31357975, 31357979,
35313131, 35313135, 35313531, 35313535, 35313575, 35313579, 35353131, 35353135, 35353531,
35353535, 35353575, 35353579, 35357531, 35357535, 35357575, 35357579, 35357975, 35357979,
35753131, 35753135, 35753531, 35753535, 35753575, 35753579, 35757531, 35757535, 35757575,
35757579, 35757975, 35757979, 35797531, 35797535, 35797575, 35797579, 35797975, 35797979,
42020202, 42020242, 42020246, 42024202, 42024242, 42024246, 42024642, 42024646, 42024686,
42420202, 42420242, 42420246, 42424202, 42424242, 42424246, 42424642, 42424646, 42424686,
42464202, 42464242, 42464246, 42464642, 42464646, 42464686, 42468642, 42468646, 42468686,
46420202, 46420242, 46420246, 46424202, 46424242, 46424246, 46424642, 46424646, 46424686,
46464202, 46464242, 46464246, 46464642, 46464646, 46464686, 46468642, 46468646, 46468686,
46864202, 46864242, 46864246, 46864642, 46864646, 46864686, 46868642, 46868646, 46868686,
53131313, 53131353, 53131357, 53135313, 53135353, 53135357, 53135753, 53135757, 53135797,
53531313, 53531353, 53531357, 53535313, 53535353, 53535357, 53535753, 53535757, 53535797,
53575313, 53575353, 53575357, 53575753, 53575757, 53575797, 53579753, 53579757, 53579797,
57531313, 57531353, 57531357, 57535313, 57535353, 57535357, 57535753, 57535757, 57535797,
57575313, 57575353, 57575357, 57575753, 57575757, 57575797, 57579753, 57579757, 57579797,
57975313, 57975353, 57975357, 57975753, 57975757, 57975797, 57979753, 57979757, 57979797,
64202020, 64202024, 64202420, 64202424, 64202464, 64202468, 64242020, 64242024, 64242420,
64242424, 64242464, 64242468, 64246420, 64246424, 64246464, 64246468, 64246864, 64246868,
64642020, 64642024, 64642420, 64642424, 64642464, 64642468, 64646420, 64646424, 64646464,
64646468, 64646864, 64646868, 64686420, 64686424, 64686464, 64686468, 64686864, 64686868,
68642020, 68642024, 68642420, 68642424, 68642464, 68642468, 68646420, 68646424, 68646464,
68646468, 68646864, 68646868, 68686420, 68686424, 68686464, 68686468, 68686864, 68686868,
75313131, 75313135, 75313531, 75313535, 75313575, 75313579, 75353131, 75353135, 75353531,
75353535, 75353575, 75353579, 75357531, 75357535, 75357575, 75357579, 75357975, 75357979,
75753131, 75753135, 75753531, 75753535, 75753575, 75753579, 75757531, 75757535, 75757575,
75757579, 75757975, 75757979, 75797531, 75797535, 75797575, 75797579, 75797975, 75797979,
79753131, 79753135, 79753531, 79753535, 79753575, 79753579, 79757531, 79757535, 79757575,
79757579, 79757975, 79757979, 79797531, 79797535, 79797575, 79797579, 79797975, 79797979,
86420202, 86420242, 86420246, 86424202, 86424242, 86424246, 86424642, 86424646, 86424686,
86464202, 86464242, 86464246, 86464642, 86464646, 86464686, 86468642, 86468646, 86468686,
86864202, 86864242, 86864246, 86864642, 86864646, 86864686, 86868642, 86868646, 86868686,
97531313, 97531353, 97531357, 97535313, 97535353, 97535357, 97535753, 97535757, 97535797,
97575313, 97575353, 97575357, 97575753, 97575757, 97575797, 97579753, 97579757, 97579797,
97975313, 97975353, 97975357, 97975753, 97975757, 97975797, 97979753, 97979757, 97979797
计数器频率 : 10000000Hz
计数器计数 : 523855
0.052386秒