参考：

<https://github.com/chenshuo/recipes/blob/master/basic/tutorial/factorial.cc>

万进制：由低到高，每4位看成十进制中的1位，进行四则运算时，产生的结果超过10000要进位，输出时最高位原样输出，其它位如果小于1000，在高位补0，1位1补。

<https://blog.csdn.net/jwg2732/article/details/51837505>

例1：



按照万进制的规则：每4位看成一个元素进行计算



每个元素分别与15相乘：

，大于10000产生进位，



，进位为1



输出为1618865580

例2：



对应的：



，大于10000产生进位，







注意：小于1000，在输出时需要在高位补0。

输出为1501265580。

大数的阶乘：

例：程序C++ Project/factoria\_test

typedef int Type;

typedef *vector*<Type> BigInt;

const Type Mod = 10000;

const double PI = 3.1415926;

const double E = 2.718281828459;

// 求10000以内的阶乘

// 超出10000有可能会出现错误

BigInt Factorial(Type n)

{

*assert*(n >= 0 && n <= 10000);

int factorial\_length = 0.5 \* *log10*(2 \* n \* PI) + n \* *log10*(n / E) + 1;

*cout* << "factorial\_length = " << factorial\_length << *endl*;

BigInt result;

result.*reserve*(factorial\_length);

result.*push\_back*(1);

for (Type factor = 1; factor <= n; ++factor)

{

Type carry = 0;

for (auto& item : result)

{

// 万进制计算

// 如果是超过10000的阶乘

// product有可能>10000 \* 10000

// 从而导致carry大于10000

Type product = item \* factor + carry;

item = product % Mod;

carry = product / Mod;

}

// 如果carry大于0，说明结果

// 超出了result现有的长度

if (carry > 0)

{

result.*push\_back*(carry);

}

}

return result;

}

// 修正万进制阶乘

// 上述的算法之所以在超过10000后

// 出现错误，是由于计算carry时会

// 出现超过10000的值，从而引发错误

// 该修正其实意义不大，如果计算超过

// 10000的阶乘，完全可以使用十万进制，

// 百万进制等

BigInt FactorialModify(Type n)

{

int factorial\_length = 0.5 \* *log10*(2 \* n \* PI) + n \* *log10*(n / E) + 1;

BigInt result;

result.*reserve*(factorial\_length);

result.*push\_back*(1);

for (Type factor = 1; factor <= n; ++factor)

{

Type carry = 0;

for (auto& item : result)

{

// 万进制计算

Type product = item \* factor + carry;

item = product % Mod;

carry = product / Mod;

}

// 如果carry大于0，说明结果

// 超出了result现有的长度

if (carry > 0)

{

// 修正

while (carry >= Mod)

{

result.*push\_back*(carry % Mod);

carry = carry / Mod;

}

if (carry > 0)

{

result.*push\_back*(carry);

}

}

}

return result;

}

void PrintBigInt(const BigInt& number)

{

if (number.*empty*())

{

*cout* << "0" << *endl*;

}

else

{

int mod\_length = *to\_string*(Mod).*size*();

*string* format = "%0" + *to\_string*(mod\_length - 1) + "d";

// 最高位按实际输出

*printf*("%d", number.*back*());

for (auto itor = number.*rbegin*() + 1; itor != number.*rend*(); ++itor)

{

// 按指定位整数输出，不足高位补0

*printf*(format.*c\_str*(), \*itor);

}

*cout* << *endl*;

}

}

int main(int argc, char\* argv[])

{

BigInt result = Factorial(10000);

PrintBigInt(result);

return 0;

}