```cpp
// constructor
BigInt(int num = 0) {
    int count = 0;
    int temp = num;
    while (temp > 0) {
        count++;
        temp /= 1000;
    }

    if (count == 0) size = 1;
    else size = count;
    digits = new int[size];

    for (int i = 0; i < size; ++i) {
        digits[i] = num % 1000;
        num /= 1000;
    }
}
```

使用temp/=1000 計算出陣列的大小
然後把0-999放進digits[i]

```cpp
// addition operator
BigInt operator+(const BigInt& other) const {
    int carry = 0, i;
    BigInt result;
    for (i = 0; (i < size || i < other.size || carry>0); ++i) {
        if (i < size) carry += digits[i];
        if (i < other.size) carry += other.digits[i];
        result.digits[i] = carry % 1000;
        carry /= 1000;
    }
    result.size = i;
    return result;
}
```

if (i < size) carry += digits[i]
將digits[i]加入進位carry

if (i < other.size) carry += digits[i]
將other. digits[i]加入進位carry

計算相加後的餘數並存入result.digits[i]

carry /= 1000
將進位carry往右移3個位元

```cpp
// multiplication operator
BigInt operator*(const BigInt& other) const {
    BigInt result;
    result.size = size + other.size;
    result.digits = new int[result.size];
    for (int i = 0; i < result.size; ++i) {
        result.digits[i] = 0;
    }
    for (int i = 0; i < size; ++i) {
        for (int j = 0; j < other.size; ++j) {
            result.digits[i+j] += digits[i] * other.digits[j];
            result.digits[i+j+1] += result.digits[i+j] / 1000;
            result.digits[i+j] %= 1000;
        }
    }
    while (result.size > 1 && result.digits[result.size-1] == 0) result.size--;
    return result;
}
```

將第一個數的每個數位與第二個數的每個數位相乘，然後再將所得的答案加起來。

兩個for loop可以確保每一個數字都有被計算。

result.digits[i+j] += digits[i] * other.digits[j]
digits[i] 和 other.digits[j] 相乘，然後將產品加到 result.digits[i+j]

result.digits[i+j+1] += result.digits[i+j] / 1000
如果result.digits[i+j] 大於等於 1000，就需要進位。
如果小於 1000，則不需要進位，直接取餘數 →result.digits[i+j] %= 1000

while 循環，當結果陣列中的最後一個元素為 0 時，它會縮小 BigInt 的大小，以避免多餘的空間浪費。

```
// output operator
friend ostream& operator<<(ostream& output, const BigInt& num) {
    output << num.digits[num.size-1];
    for (int i = num.size - 2; i >= 0; --i) {
        output<<",";
        output.width( wide: 3);
        output << num.digits[i];
    }
    return output;
}
```

Output 從num.size-1開始，令”,”不會出現在最前面

For loop從num.size-2開始，因為num.size-1已儲存在output
列印” , ”
output.width確保長度是3
然後ouput digit[i]