ARPREC 软件包使用总结

说明：高精度软件包 ARPREC 是由加州大学和劳伦斯伯克利国家实验室开发，通过使用 double 型数组来存储高精度数据的各部分来实现高精度计算的开源包，支持 C++和 fortran 语言。但是该软件缺乏相关文档和使用手册，使得用户的上手使用变得困难。本人在阅读部分源码并改进和实际使用之后，总结出以下基本的不完全使用说明，如有问题欢迎随时指正，联系方式：zjqucas@gmail.com。Copyright 所有权和最终解释权归该软件原作者。

一、变量声明和初始化

ARPREC 提供了三种基础变量类型，mp_real, mp_int, mp_complex，分别对应实数，整数和复数。下面以常用的实数类型为例说明。

实数声明一般直接使用 mp_real 类型来定义，需要引入头文件<arprec/mp_real.h>,其他类似。初始化方法则有很多种，源码定义了六种之多，归纳起来有三大种：从浮点数初始化，从字符串初始化，从高精度类型初始化。其中前两种初始化均需要指定精确度。可参考如下范例：

```
mp_real ();
mp_real (double a,int size);
mp_real (char *s,int size);
mp_real (string s,int size);
mp_real (mp_real a);
```

这里我们建议从字符串中初始化，方便消息通信的数据传递。其他类型的数据初始化和 mp_real 类似。

二、高精度运算

ARPREC 的每一种数据类型都对常见运算符做了重载，包括 + ，- ，* ，/，++ ，-- ，+= ，>，<，>=等等，你可以在高精度数据类型之间无缝使用这些运算符就像使用常规数据类型。除此之外，ARPREC 还重载了一些常用数学函数，比如 pow,sin,cos,log,log10,asin,sqrt,abs 等 ,你也可以直接使用，这里不全部列举，具体可以参考相关头文件。需要注意的是，在常规 c++语言中不同数据类型之间的运算会发生隐式类型转换，但高精度类型在和不同类型如 double 类型数据混合运算则会报错，因此在计算时应尽量保证同类型的数据之间进行。另外 ARPREC 计算支持同类型不同精度格式之间的运算，下面给出如下范例：

```
m_real a =1.23343345345234234567567;
mp_real b=3.1415926;
mp_real c =a+b;
pow( a,b);
```

三、输入和输出

在开始使用高精度类型之前，需要提前定义数据精度初始化 ARPREC 包，否则编译器会报错未初始化。初始化的函数是：mp :: mp_init(int new_digits),其中 n_words 是精确度参数。结束也需要调用 mp::mp_finalize()。ARPREC 默认支持多种数据格式输入，包括科学计数法和全精度输入。如

```
10 ^ 3 x 1.23423425231232343435,
2342.12314324324123324,
123142e1200
```

等，多个数据之间需要用逗号隔开，两个数之间的所有空格读取时会被忽略。从

外部文件读取数据需要使用流，再配合 mp_real 类的 read 方法。同样的，向外部文件写数据也需要使用流，再配合 mp_real 类的 write 方法。输出数据建议配合 cout.precision(OUT_PRECISION)规范输出格式。这里推荐使用科学计数法的格式表示数据，方便流的读取。下面给出范例(假设 file 文件里有一个 mp_real 类型数据)：

```
mp::mp_init(200);
mp_real  a ;
fstream stream;
stream.open(“file.txt”,ios::in|ios::out);
a.read(stream);
a.write(stream,200,0,2);
mp::mp_finalize();
```

四、MPI 通信

由于 mpi 框架支持有限的基本数据类型，不支持此类高精度的数据类型，并行计算在多节点通信时传输数据会变得不方便。如我在上文中提到的，建议从字符串中初始化数据，所以在这里我建议在传递高精度数据类型前先把数据转为字符串之后再传输，已达到节省空间和方便传输的目的。接收方再进行解码的的操作获得正确的数据。下面给出字符串转换的示例：

```
mp_real  a = mp_real(“1.34564656768678979”);
string  str_a = a.to_string( );
const  char *p = str_a.c_str( );
char  *ans = const_cast<char*> (p);
```

五、实例

下面附上一个矩阵相乘的实例。

```
fstream  file_A,file_B,file_C;
file_A.open(“A.txt”,ios::in);
file_B.open(“B.txt”,ios::in);
file_C.open(“C.txt”,ios::out);
mp_real A[100][100],B[100][100],C[100][100]; //定义 mp_real 类型数组
for (int  i=0 ; i<100 ; i++) {
    for(int  j=0 ; j<100 ; j++){
        A[i][j].read(file_A) ;
        B[i][j].read(file_B) ;//读取数据初始化数组
    }
}
for( int i=0 ; i<100 ; i++){
    for(int j=0 ; j<100; j++){
        for(int k=0;k<100;k++){
            C[i][j] += A[i][k] * B[k][j] ; //高精度计算
        }
    }
}
for(int i=0 ; i<100 ; i++){
    for(int j=0;j<100;j++){
        C[i][j].write(file_C,30,0,2); //写出到文件保留 30 位精度
        file_C << “,” ; //必须加入逗号隔开每两个数
    }
}
```