西北工业大学
NORTHWESTERN POLYTECHNICAL UNIVERSITY

C程序设计

Programming in C

1011014

主讲：姜学锋，计算机学院

函数之间实现批量数据传递

- 3、高精度运算算法
- 4、与Matlab混合编程

6.3.2 数组参数的传递机制

高精度运算库GMP

GMP
«Arithmetic without limitations»

GMP全称为GNU Multiple Precision Arithmetic Library，即GNU高精度算术运算库，http://gmplib.org/

GMP功能非常强大，接口很简单，文档详细，既有C风格的接口，也有C++精心封装后的接口。

6.3.2 数组参数的传递机制

高精度运算库GMP

GMP 是一款高精度计算、符号整数、有理数和浮点数操作的数学计算库。该库没有精度的限制，除了隐式的可用内存。它拥有一套丰富的函数，其目的是为小型或巨大的运算对象提供更快速的运算，通过使用大量CPU最常见的内循环优化的汇编代码的快速算法。

GMP不但有普通的整数、实数、浮点数的高精度运算，还有随机数生成，尤其是提供了非常完备的数论中的运算接口，比如Miller-Rabin素数测试算法、大素数生成、欧几里德算法、求域中元素的逆、Jacobi符号、legendre符号等。

GMP本身提供了很多例子程序，很容易将它们集成到自己的代码中去。

6.3.2 数组参数的传递机制

高精度运算库GMP

如何制作GMP GCC库？

GMP官方网站没有提供直接的GCC库可使用，而是提供了源码。下面介绍由源码，自行产生GCC库的方法。

（1）下载GMP源码，例如：gmp-6.1.0.tar.bz2和使用手册gmp-man-6.1.0.pdf。

（2）编译产生GCC库需要下载MSYS工具，

http://downloads.sourceforge.net/mingw/MSYS-1.0.11.exe

然后安装MSYS

6.3.2 数组参数的传递机制

高精度运算库GMP

（3）安装MSYS过程中，注意设置MingW编译器目录，例如：C:\Dev\CodeBlocks\mingw

（4）解压缩GMP源码，例如：C:\Dev\gmp。

（5）运行MSYS.bat，进入到C:/Dev/gmp。注意必须使用/表示目录间隔，因为MSYS命令是Linux格式。

（6）输入./configure命令执行，开始自动配置编译参数。

（7）执行make命令，开始编译。

6.3.2 数组参数的传递机制

高精度运算库GMP

（8）编译完成后产生了库文件.a。

（9）把.a库文件复制系统LIB路径中，把gmp.h复制系统INCLUDE路径中。

（10）现在，可以在工程文件添加GMP GCC库了。

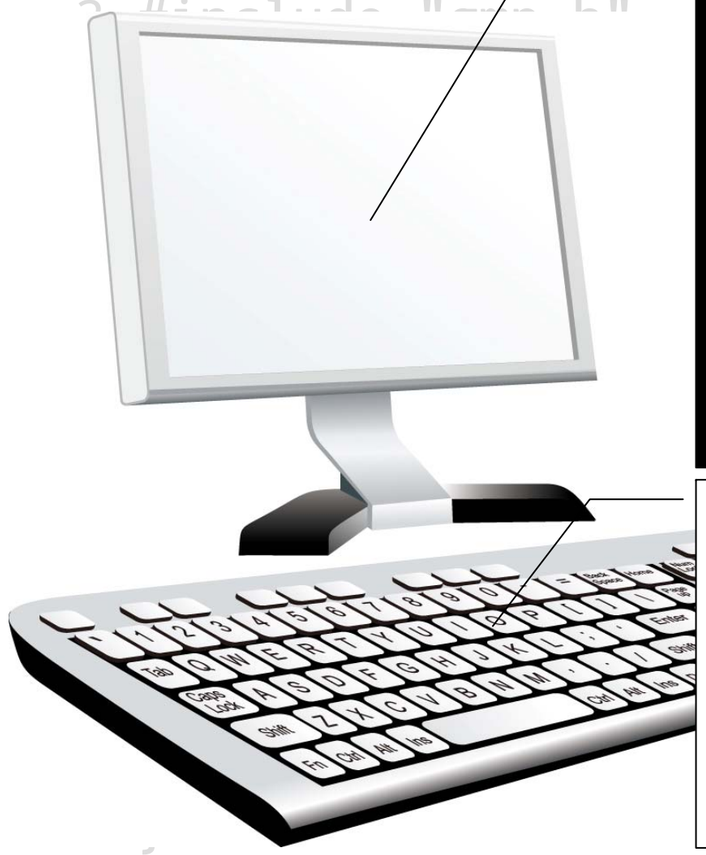
6.3.2 数组参数的传递机制

例6.64

```
#include <stdlib.h>
#include <stdio.h>
#include "gmp.h"
int main ()
{ char inputstr[3000];
  mpf_t sq_me, sq_out, test;
  mpf_set_default_prec(10000);//默认精度
  mpf_init(sq_me); mpf_init(sq_out);  mpf_init(test);
  gets(inputstr);//按字符串输入大数
  mpf_set_str (sq_me, inputstr, 10);
  mpf_sqrt(sq_out, sq_me); //对sq_me求平方根sq_out,
  mpf_mul(test,sq_out,sq_out); //test=sq_out*sq_out
  gmp_printf ("Square root: %.200Ff\n\n", sq_out);
  gmp_printf ("Re-squared : %Ff\n\n", test);
  return 0;
```

6.3.2 数组参数的传递机制

```
1 #include <stdlib.h>
2 #include <stdio.h>
```



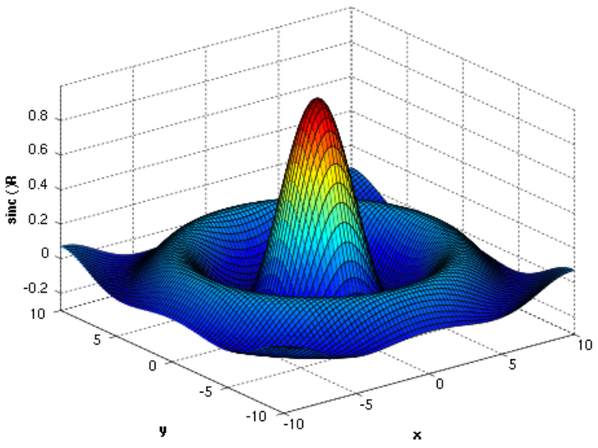
```
Square root:
4952238331303110980924222615988628334869566046038127132471492868065
4813093947239963401678377595561892102 8.19202568258368255653837168412
9235643266154861433201410617463895139059667295039498109899238811630
8883326004535647648563996144250924277757344248059826024201642748515
3256554388985581780728209159072289 0002
Re-squared :
2452466449002782119765176635730880184670267876783327597434144517150
6160083003858721695220839933207154910362682719167986407977672324300
5600592035631246561218465817904100131859299619933817012149335034875
870551067.000000
```

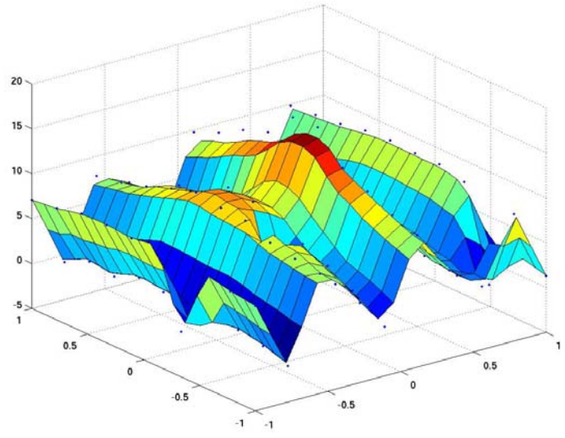
```
2452466449002782119765176635730880184670267876783327597434144517150
6160083003858721695220839933207154910362682719167986407977672324300
5600592035631246561218465817904100131859299619933817012149335034875
870551067↙
```

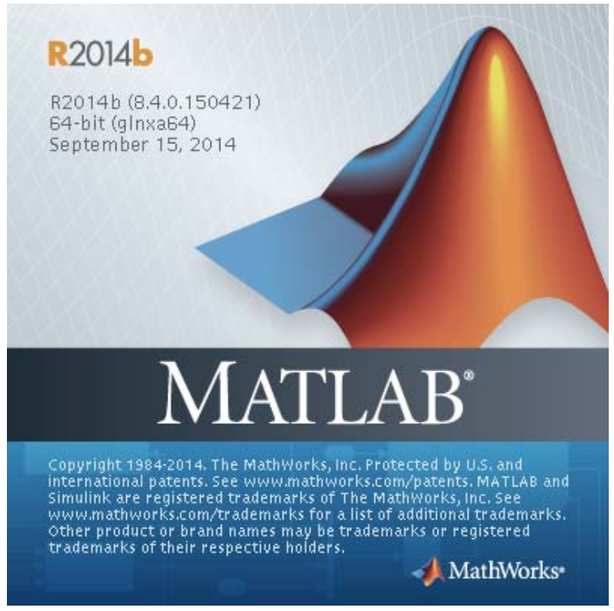
6.3.2 数组参数的传递机制

与面向科学计算的MATLAB混合编程

MATLAB是美国MathWorks公司出品的商业数学软件，用于算法开发、数据可视化、数据分析以及数值计算的高级技术计算语言和交互式环境，主要包括MATLAB和Simulink两大部分。







6.3.2 数组参数的传递机制

与面向科学计算的MATLAB混合编程

MATLAB意为矩阵工厂（矩阵实验室）将数值分析、矩阵计算、科学数据可视化以及非线性动态系统的建模和仿真等诸多强大功能集成在一个易于使用的视窗环境中，为科学研究、工程设计以及必须进行有效数值计算的众多科学领域提供了一种全面的解决方案，并在很大程度上摆脱了传统非交互式程序设计语言（如C、Fortran）的编辑模式，代表了当今国际科学计算软件的先进水平。

6.3.2 数组参数的传递机制

与面向科学计算的MATLAB混合编程

MATLAB软件有用于C/C++的外部程序接口，简称MATLAB API。通过这个接口，数据可以在MATLAB与C/C++程序之间相互传递。从而使C/C++程序能够利用MATLAB在数值分析、矩阵计算、信号处理和图形显示等领域的优势，弥补其在科学计算、显示数据图形方面的不足，让程序员能够进行科学计算、数据仿真等开发工作。

6.3.2 数组参数的传递机制

与面向科学计算的MATLAB混合编程

C/C++与MATLAB是通过组件对象模型COM（component object model）相互通信的，因而MATLAB API调用有多种方法。下面介绍通过调用MATLAB引擎来实现C/C++与MATLAB混合编程的方法。

6.3.2 数组参数的传递机制

与面向科学计算的MATLAB混合编程

1．MATLAB引擎

所谓MATLAB引擎（engine）是指一组MATLAB提供的接口函数，支持C/C++、Fortran等编程语言。通过这些接口函数，程序员可以在编程语言环境中实现对MATLAB的控制。主要功能有：

打开/关闭一个MATLAB对话。

向MATLAB环境发送命令字符串。

从MATLAB环境中读取数据。

向MATLAB环境中写入数据。

6.3.2 数组参数的传递机制

与面向科学计算的MATLAB混合编程

与其他接口方法相比，引擎所提供的MATLAB功能支持是最全面的。通过引擎方式，应用程序会打开一个新的MATLAB进程，可以控制它完成任何计算和绘图操作，且对所有数据结构提供完全的支持。同时，引擎方式打开的MATLAB进程会在任务栏显示自己的图标，在该窗口中可以观察主程序控制MATLAB运行的流程，并且可以在其中输入任何MATLAB命令。

6.3.2 数组参数的传递机制

与面向科学计算的MATLAB混合编程

通过引擎方式建立的对话是将MATLAB以ActiveX控件方式启动的，在MATLAB初次安装时会自动执行一次：

```
matlab /regserver
```

6.3.2 数组参数的传递机制

- 将MATLAB注册到系统的控件库中。如果因为特殊原因无法打开MATLAB引擎，可以在命令行提示符中执行这个命令重新注册。

6.3.2 数组参数的传递机制

- 下面给出MATLAB引擎的接口函数说明，其头文件为engine.h。

6.3.2 数组参数的传递机制

- （1）engOpen函数

```
函数原型：Engine *engOpen(const char *startcmd);
```

- engOpen启动MATLAB引擎会话。startcmd是用来启动MATLAB引擎的字符串参数，在Windows操作系统中只能为NULL。函数返回值是一个Engine结构类型的指针，它与文件指针类似。在其他MATLAB引擎接口函数中，都要用到这个指针。如果启动MATLAB引擎失败，函数返回NULL。

6.3.2 数组参数的传递机制

- （2）engOpenSingleUse函数

```
Engine *engOpenSingleUse(const char *startcmd,void *dcom,i
nt *retstatus);
```

- engOpenSingleUse与engOpen类似。dcom为NULL，retstatus为一个整型变量的指针，按地址传递方式得到返回状态。engOpenSingleUse与engOpen不同的是它用一个独立的COM启动MATLAB引擎会话，此COM与其他是不共用的。因此多个engOpenSingleUse调用启动了多个引擎会话，即engOpenSingleUse可以使用多进程MATLAB引擎。

6.3.2 数组参数的传递机制

- （3）engClose函数

```
函数原型：int engClose(Engine *ep);
```

- engClose关闭MATLAB引擎会话。ep为已打开Engine指针，不能两次关闭ep。

6.3.2 数组参数的传递机制

- （4）engOutputBuffer函数

```
函数原型：int engOutputBuffer(Engine *ep,char *p,int n);
```

- MATLAB执行命令后会在窗口中输出信息，engOutputBuffer用来指令MATLAB将输出信息保存到p所指向的存储空间。ep为已打开Engine指针，p为一个字符足够多的字符数组，n为最大保存的字符个数。如果要停止保存，只需调用：

```
engOutputBuffer(ep,NULL,0);
```

6.3.2 数组参数的传递机制

- （5）engEvalString函数

```
函数原型：int engEvalString(Engine *ep,const char *string);
```

- engEvalString发送一个命令让MATLAB执行。ep为已打开Engine指针，string为MATLAB命令字符串。

6.3.2 数组参数的传递机制

- （6）engSetVisible函数

```
函数原型：int engSetVisible(Engine *ep,bool value);
```

- 默认情况下，以引擎方式启动MATLAB时，会打开MATLAB主窗口，可在其中随意操作，但有时也会干扰应用程序的运行。engSetVisible用来显示或隐藏主窗口，ep为已打开Engine指针，value为1表示显示主窗口，为0表示隐藏主窗口。

6.3.2 数组参数的传递机制

- （7）engGetVisible函数

```
函数原型：int engGetVisible(Engine *ep,bool *value);
```

- engGetVisible获取ep引擎主窗口的显示/隐藏方式，value为指向bool类型变量的指针，按地址传递方式得到显示或隐藏方式值（1表示显示，0表示隐藏）。

6.3.2 数组参数的传递机制

- （8）engPutVariable函数

```
函数原型：int engPutVariable(Engine *ep,const char *name,co
nst mxArray *pm);
```

- engPutVariable向MATLAB引擎工作空间写入变量。ep为已打开Engine指针，mp为指向被写入变量的指针，name为变量写入后在MATLAB工作空间中的变量名。

6.3.2 数组参数的传递机制

- （9）engGetVariable函数

```
函数原型：mxArray *engGetVariable(Engine *ep,const char *name);
```

- engGetVariable从MATLAB引擎工作空间中获取变量。ep为已打开Engine指针，mp为指向被写入变量的指针，name为工作空间中的变量名，函数返回值指向获取变量。

6.3.2 数组参数的传递机制

- 2．MATLAB数据类型
- MATLAB引擎函数中，所有与变量有关的数据类型都是mxArray类型。mxArray是一种复杂的数据结构，与MATLAB的array类型相对应，C/C++和MATLAB的数据交互都是通过mxArray来实现的。mxArray类型对应的头文件为matrix.h，由于在engine.h文件里面已经包含了matrix.h，因此程序无需重复包含。

6.3.2 数组参数的传递机制

- mxArray的建立采用mxCreatexxx形式的函数，例如新建一个double类型数组，可用函数mxCreateDoubleMatrix，函数原型为：

```
mxArray *mxCreateDoubleMatrix(mwSize m,mwSize n,mxComplexity ComplexFlag);
```

- m和n为矩阵的行和列。ComplexFlag为常数，用来区分矩阵中元素是实数还是复数，取值分别为mxREAL和mxCOMPLEX。

6.3.2 数组参数的传递机制

- 例如下面程序代码创建了一个3行5列的二维实数mxArray数组：

```
mxArray *T;
T=mxCreateDoubleMatrix(3,5,mxREAL);
```

6.3.2 数组参数的传递机制

- 对应的，要删除一个mxArray数组使用mxDestroyArray函数，原型为：

```
void mxDestroyArray(mxArray *pm);
```

6.3.2 数组参数的传递机制

- pm为要删除的数组指针。例如要删除上面创建的数组T，代码如下：

```
mxDestroyArray(T);
```

- 类似的创建函数还有：

```
mxArray *mxCreateString(const char *str);
```

- 创建一个字符串类型并初始化为str字符串。

6.3.2 数组参数的传递机制

- 对于常用的double类型数组，可以用mxGetPr和mxGetPi两个函数分别获得其实部和虚部的C/C++数据指针，两个函数原型为：

```
double *mxGetPr(const mxArray *pm); //返回数组pm的实部指针
double *mxGetPi(const mxArray *pm); //返回数组pm的虚部指针
```

6.3.2 数组参数的传递机制

- 这样C/C++就可以通过获得的指针对mxArray数组中的数据进行读写操作。例如可以用函数engGetVariable从MATLAB引擎工作空间读入mxArray数组，然后用mxGetPr和mxGetPi获得数据指针，对其中的数据进行C/C++方式的处理，最后调用engPutVariable 函数将修改后的数组重新写入到MATLAB引擎工作空间。
- MATLAB API还有很多函数，可以通过使用手册去查阅。

6.3.2 数组参数的传递机制

- 3．在Visual C++ 6.0中调用MATLAB引擎
- 使用前面的“添加路径法”配置开发环境。设MATLAB安装文件夹（例如C:\Program Files\MATLAB）为<MATLAB>，将<MATLAB>\extern\include路径添加到VC6的系统INCLUDE路径中，将<MATLAB>\extern\lib\win32\microsoft路径添加到VC6的系统LIB路径中。根据程序中使用的MATLAB API函数来确定包含哪些头文件和连接库

6.3.2 数组参数的传递机制

- 例如：

```
#include <mex.h>      //MATLAB接口
#include <engine.h> //MATLAB ENGINE接口
#pragma comment(lib, "libeng.lib") //MATLAB ENGINE库
#pragma comment(lib, "libmx.lib")  //MATLAB API库
#pragma comment(lib, "libmex.lib") //MATLAB库
#pragma comment(lib, "libmat.lib") //MATLAB MAT-File库
```

6.3.2 数组参数的传递机制

【MATLAB混合编程举例】

调用MATLAB绘制图形。

6.3.2 数组参数的传递机制

例6.67

```
#include <stdio.h>
#include <string.h>
#include <engine.h> //MATLAB ENGINE接口
#pragma comment(lib, "libeng.lib") //MATLAB ENGINE库
#pragma comment(lib, "libmx.lib")  //MATLAB API库
int main()
{
   Engine *ep;
   mxArray *T = NULL;
   double P[10]={0.0,1.0,2.0,3.0,4.0,5.0,6.0,7.0,8.0,9.0 };
   if ((ep=engOpen(NULL))==NULL) { //打开MATLAB引擎
      printf("MATLAB Engine fail\n");
      return -1;
   }
   engSetVisible(ep,0); //隐藏MATLAB主窗口
```

6.3.2 数组参数的传递机制

例6.67

```
16    T=mxCreateDoubleMatrix(1,10,mxREAL); //创建矩阵
17    memcpy((void*)mxGetPr(T),(void*)P,sizeof(P));
18    //复制数组P到矩阵中
19    engPutVariable(ep,"T",T); //将矩阵送到MATLAB中
20    engEvalString(ep,"D = .5.*(-9.8).*T.^2;"); //MATLAB命令
21    printf("figure1 : MATLAB Plot\n");
22    //MATLAB绘制图形
23    engEvalString(ep,"figure1=figure('Color',[1 1 1]);");
24    //MATLAB图形窗
25    engEvalString(ep,"plot(T , D);"); //MATLAB绘制图形
26    engEvalString(ep,"title('Position/Time');");
27    //MATLAB设置图形标题
28    engEvalString(ep,"xlabel('Time(s)');");//MATLAB设置x轴标签
29    engEvalString(ep,"ylabel('Position(m)');");
30    //MATLAB设置y轴标签
```

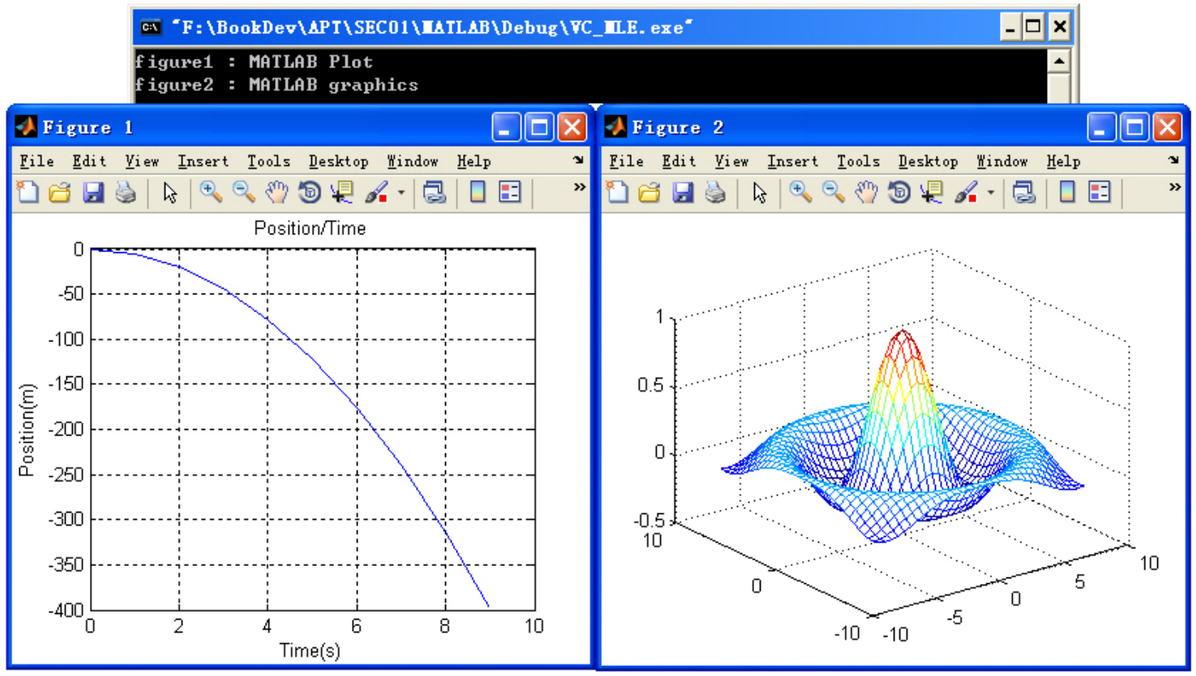
6.3.2 数组参数的传递机制

例6.67

```
31    engEvalString(ep,"grid;"); //MATLAB绘制坐标
32    printf("figure2 : MATLAB graphics\n");//MATLAB绘制mesh图形
33    engEvalString(ep,"figure2=figure('Color',[1 1 1]);");
34    engEvalString(ep,"[X,Y] = meshgrid(-8:.5:8);");
35    engEvalString(ep,"R = sqrt(X.^2 + Y.^2) + eps;");
36    engEvalString(ep,"Z = sin(R)./R;");
37    engEvalString(ep,"mesh(X,Y,Z);");
38    getchar(); //暂停
39    mxDestroyArray(T); //释放矩阵
40    engClose(ep); //关闭MATLAB引擎
41    return 0;
42 }
```

6.3.2 数组参数的传递机制

- 程序运行结果如图所示。




程序设计