# 第八次上机作业

目录

[第八次上机作业 1](#_Toc405848047)

[work1：函数练习：函数的定义和调用 2](#_Toc405848048)

[work2：全局变量练习：一元二次方程求根 3](#_Toc405848049)

[work3：多文件工程：多人协作 3](#_Toc405848050)

[work4：多文件工程：多人协作 3](#_Toc405848051)

[work5：指针变量基本概念：指针变量及指针的指向 3](#_Toc405848052)

[work6：指针变量和函数：利用指针改变主调函数中的变量内容 4](#_Toc405848053)

[work7：指针变量和数组：指针和数组的关系 5](#_Toc405848054)

[work8：指针变量和数组：指针和数组的关系 6](#_Toc405848055)

[work9：指针变量和数组：指针和数组的关系 6](#_Toc405848056)

[work10：指针变量和数组：指针和数组的关系 6](#_Toc405848057)

编程建议：在提供函数时，养成注释函数的习惯，良好的注释对代码的维护、复用有**非常大**的帮助：在函数头之前加一段注释，解释函数名、函数形参、函数返回值、函数的作用、函数的版本、使用函数的注意点、函数作者email等信息，如下是一个函数头注释的模板：

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
\*\* Function name: F()  
\*\* Input argv:  
\*\*      --    
\*\* Return:  
       true: success  
       false: fail  
\*\* Function Description:

\*\* Version: 1.0  
\*\* Be careful:

\*\*Change log:

\*\*Author:  
\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

如对求根函数root的注释说明如下：

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
\*\* Function name: root  
\*\* Input argv:  
      double a,b,c: the coefficient of the quadratic equation with one unknown  
\*\* Return:  
       int: the number of the roots.

2: the equation has two roots.

1: the equation has one roots.

0: the equation has no root.  
\*\* Function Description:

This function can find the roots of quadratic equation with one unknown.

\*\* Version: 1.0  
\*\* Be careful:

The results are saved in two global vars named g\_x1 and g\_x2

\*\*Change log:

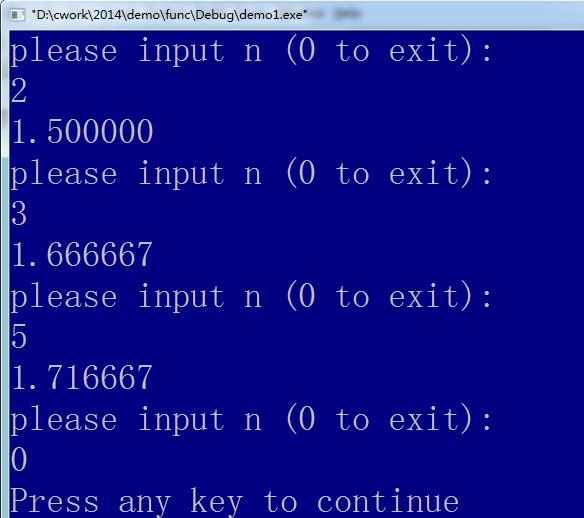
Created in 20141208

\*\*Author:

shzhang@mail.xidian.edu.cn  
\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

work1：函数练习：函数的定义和调用

编写函数int factor(int n)，用于求解n!，编写函数double sum\_factor(int n)，用于求解，在主函数中接收一个小于13的n值，调用sum\_factor函数，并输出计算结果（考虑一下，为何这里要有小于13的限制？）。



work2：全局变量练习：一元二次方程求根

编写函数int root(double a,double b,double c)，用于求解一元二次方程的根x1和x2，x1和x2定义为全局变量，在root函数中赋值，在main函数中输出，root函数的返回值为根的个数。需要提供三个测试例：

a=1，b=2，c=3时，没有根；

a=1，b=3，c=2时，两个根：-1和-2

a=1，b=4，c=4时，一个根：-2

work3：多文件工程：多人协作

学号为奇数的同学：

在f1.c中，定义函数double tri\_area(double a, double b,double c)，计算给定三角形的面积，并通过返回值将计算结果带回：如果输入的三条边不能构成三角形，则返回-1；如果能够构成三角形，则返回三角形的面积。

在f1.h中，给出函数的声明。

学号为奇数+1的偶数同学：

得到f1.c和f1.h，在main.c中，完成main函数，接收键盘输入的三条边的值，调用f1.c中的tri\_area函数，输出对应三角形的面积。

work4：多文件工程：多人协作

学号为偶数的同学：

在f1.c中，定义函数int is\_perfect\_num(int x)，用于判断整数x是否为完数（关于完数的定义，请参考第五次上机作业），如果是，则返回真，否则返回假。

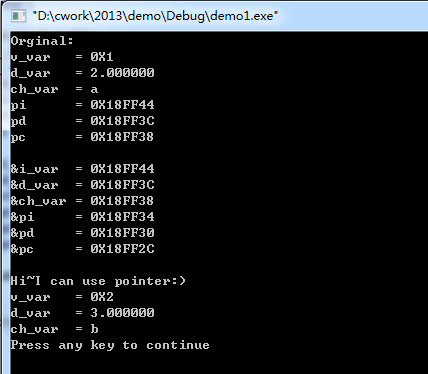
在f1.h中，给出函数的声明。

学号为偶数-1的同学：

得到f1.c和f1.h，在main.c中，完成main函数，通过调用is\_perfect\_num函数，输出1000以内的所有完数。

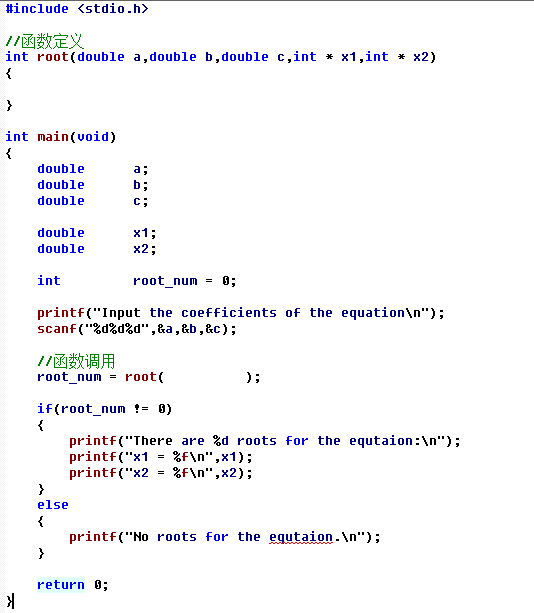
work5：指针变量基本概念：指针变量及指针的指向

定义一个整型变量i\_var，双精度浮点型变量d\_var，字符变量ch\_var，分别初始化为1，2.0，和’a’，定义三个指针变量pi，pd，pc，用于存放i\_var、d\_var以及ch\_var的地址，用%#X格式打印六个变量的值和它们在内存中的地址（%#表示的输出提示方式，如果是8进制，在前面加0，如果是十进制，不加任何字符，如果是十六进制，会加上0x）（如果变量是指针变量，对应的输出格式也经常会使用%p），再通过指针变量pi,pd和pc，将i\_var、d\_var以及ch\_var的值改变为2,3.0和’b’。根据课件所示，画出对应变量在内存中的存储地址以及对应的内容（提交doc文档，不同机器上的运行结果可能不同）。（小提示：在printf函数中试一下\t格式吧☺，它会让你的输出更美貌）。



work6：指针变量和函数：利用指针改变主调函数中的变量内容

在work2中，我们编写了函数int root(double a,double b,double c)，用于求解一元二次方程的根，由于C函数只能带回一个返回值，因此我们用两个全局变量x1和x2记录方程的根，由于x1和x2是全局变量，因此它们对root函数和main函数均可见，我们可以在root函数中对其进行赋值，在main函数中对其进行输出。但是这种做法并不推荐，全局变量带来若干不便（在提供root函数时，必须同时提供x1和x2，如果使用函数的人，也定义了全局变量x1和x2，该怎么办？由于全局变量对全局可见，其它地方有可能会对其进行读写，有可能会对计算结果造成不利影响），因此我们更加推荐使用指针带回多个函数计算结果。重新实现函数int root(double a,double b,double c,int \* x1,int \* x2) 求解一元二次方程的根：



在这种方式中，记录根的变量x1和x2，是main函数中的局部变量，因此对root函数不可见，root函数无法通过直接访问的方式为它们赋值。我们可以通过将它们的地址传递给root函数的形参，在root函数中，利用间接访问的方式，改变x1和x2的值。请完成这个函数（函数的定义以及函数的调用，提示，想想我们在swap例程中是怎么做的？）同时，参考课件上swap函数的例程，画出main函数中root函数调用前、root函数中以及main函数中root调用后内存的分布图例（提交doc文档），同时回答，相对全局变量的情况，这种方式带来什么好处？（另外思考，作为函数的提供者，如果函数的调用者（不慎或者恶意），在调用函数时，给出了非法的实参，如指向NULL的指针变量，那么函数的提供者该如何防止自己的代码崩溃？对于函数实现者而言，无法保证函数调用者一定会给出一个合法的参数，所以必须要进行参数检查，如果参数非法，需要提前终止函数的执行，并且通过错误返回值提示函数调用者）

work7：指针变量和数组：指针和数组的关系

定义一个长度为10的整型数组int i\_arr={0,1,2,3,4,5,6,7,8,9}，用一个存放整型变量的整型指针pi，使其存放数组i\_arr的首地址，编写程序，完成下表（提交源文件&doc文档）；如果pi的初始值为i\_arr + 3，重新填写上表（提交doc文档）

|  |  |  |  |
| --- | --- | --- | --- |
| 表达式 |  | 表达式 |  |
| i\_arr |  | Pi |  |
| i\_arr + 4 |  | pi + 4 |  |
| i\_arr[4] |  | pi[4] |  |
| \*i\_arr + 4 |  | \*pi + 4 |  |
| \*(i\_arr + 4) |  | \*(pi + 4) |  |
| i\_arr[-2] | 非法 | \*(pi – 2) | 非法 |
| &i\_arr |  | &pi |  |
| &i\_arr[4] |  | &pi[4] |  |
| &i\_arr + 4 |  | &pi + 4 |  |
| &i\_arr[-2] |  | pi – 2 |  |

work8：指针变量和数组：指针和数组的关系

定义一个长度为5的整型数组int arr[] = {3,1,4,5,2}，利用指针对其进行操作，使其按照降序排列（排序算法可参考课本选择排序法代码，只不过所有的查找、交换、移位的操作，都需要利用指针完成，work9、10亦同）。

work9：指针变量和数组：指针和数组的关系

定义一个长度为5的整型数组int arr[] = {1,2,3,4,5}，利用指针对其进行操作，在不引入新数组的前提下，使其逆序排列（逆序算法参考课本逆序例题）。

work10：指针变量和数组：指针和数组的关系

定义一个长度为80的字符型数组char str[80]，接收用户从键盘输入的一个字符串，利用指针对其进行操作，在不引入新数组的前提下，使其逆序排列。如用户输入“abcdefg”，利用指针对字符串进行操作后，使得puts(str)时，输出gfedcba（考虑一下如果操作对象是字符数组中的一个字符串，和整型数组有何不同？）。