

09 | 函数：自己动手实现低配版 scanf 函数

2020-01-25 胡光

人人都能学会的编程入门课

[进入课程 >](#)



讲述：胡光

时长 23:48 大小 19.08M



你好，我是胡光，今天是大年初一，是咱们中国传统的重要节日，春节！能在这样的节日氛围里，还能坚持过来学习的，我必须要说一声“鼠”你最优秀！在这里我也祝福热爱学习的你，在新的一年里，身体健康，阖家欢乐！

今天呢，我们的学习课程也将迎来里程碑式的一课。所谓里程碑，是因为在这一节之前，你写的程序，只是片段，只是思想的随意表达，而通过了本节的学习，你的程序结构将发生翻天覆地的变化，会变得规格严整，变得可以复用，变得易于找错。



前面的课程，我们主要就是在一些基本的程序结构中做学习，包括顺序结构，分支结构以及循环结构。今天这一节中，我们将要认识的函数，可以将功能封装成可以复用的模块，就像创造乐高积木一样，废话不多说，开始今天的学习吧。

今日任务

对程序的输入输出函数，你应该已经很熟悉了。今天我们仿照 scanf 函数，实现一个低配版的 my_scanf 函数。这个函数的功能，简单来说就是将一个字符串信息转换成整型数字，能够完成这个任务，你会更深刻的理解 scanf 函数，更深刻的理解参数设计。下面给你几个例子。


首先先来看第一个基础功能：

 复制代码

```
1 int n = 98;
2 my_scanf("12345", &n);
3 printf("%d", n); // 输出 12345，而不是 98
```

上面这段代码中，我们利用 my_scanf 函数，将字符串信息转换成了整型数据，并且将结果存储到了 n 变量的内存空间中，调用 printf 函数打印 n 变量值的时候，输出的信息不是 n 变量原有的初值 98，而是 12345。对于这个基础的转换功能，要考虑兼容负数的情况。

只有这一个基础功能肯定是远远不够的，下面就让我们看另外一种情况：

 复制代码

```
1 int n = 98, m = 0;
2 my_scanf("123 45", &n, &m);
3 printf("n = %d m = %d", n, m); // 输出 n = 123 m = 45
```

上面这段代码中，首先我们定义了两个整型变量 n 和 m，然后给 n 初始化为 98，m 初始化为 0。之后给 my_scanf 函数传入的字符串信息中有一个空格，那么 my_scanf 函数会以空格作为分隔符，将第一个转换出来的数字 123 赋值给 n，第二个转换出来的数字 45 赋值给 m。

上面举例了 my_scanf 函数转换 1 个整型参数和 2 个整型参数情况，这些都是在函数的基本知识范围内的内容。经常有初学者学完函数相关的基本知识以后，就认为自己掌握了函数的全部知识，但事实绝非如此，而之所以初学者有这种“假想”，是因为他不知道如何找到和判定自己的知识盲区。

所以今天我们要讲的内容就是破除“假想”。这个任务就是要设计一个能够转换任意个整型参数的 `my_scanf` 函数，注意，这里的重点难点，可是在任意个参数上面。清楚了任务以后，下面就让我们进入今天的查缺补漏环节吧。

必知必会，查缺补漏

要完成今天的这个任务，首先你需要知道如何实现一个基本的函数，由于要支持转换任意多个整型参数，所以你还需要知道变参函数相关的知识。下面我们就逐项的来进行学习吧。

1. 函数的基础知识

数学中的函数，大家都不陌生，一般的形式是 $f(x) = y$ ， x 是自变量， y 是函数值。程序中的函数，和数学中的函数基本一致，有自变量，我们称作“传入参数”，还有函数值，我们叫做返回值。

先让我们来看一下程序中的函数的基本组成部分：

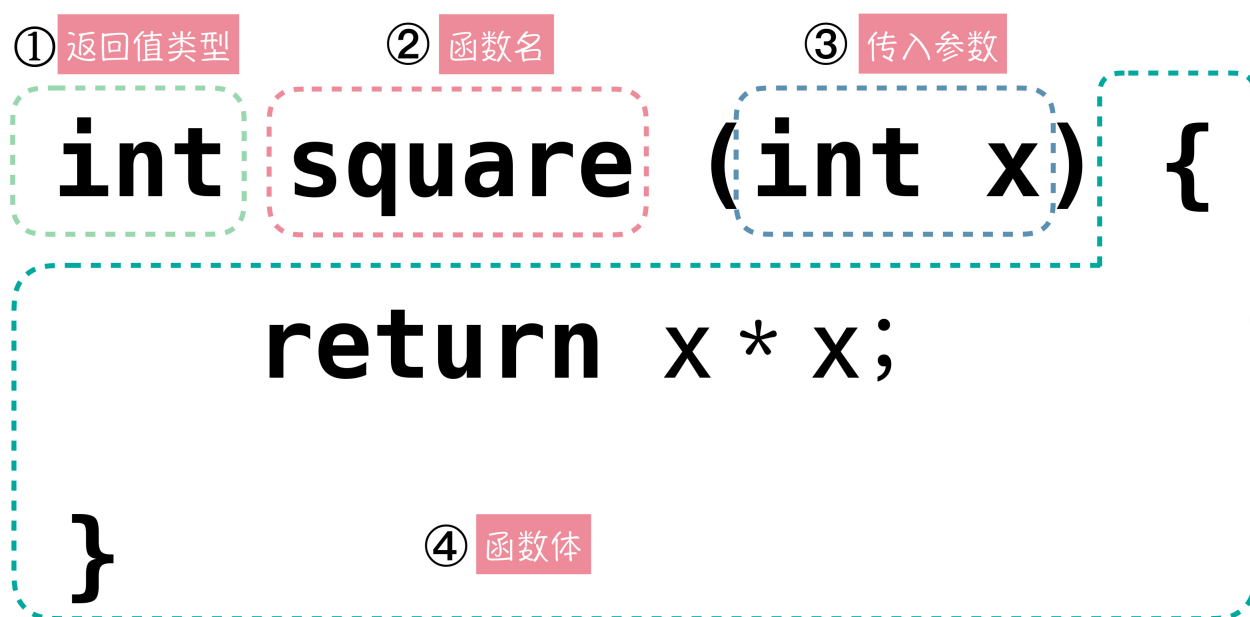


图1：函数的基本组成部分

如图 1 所示，一个程序函数从左到右，从上到下，大体可以分成四个部分：

第一个部分，是函数返回值的类型。

第二个部分，是函数名字，调用函数的时候，需要给出这个函数名，所以在设计函数名的时候，要尽量设计一个与函数功能有关的名字，例如上图中的函数，通过名字我们可知，这就是一个求平方的函数。

第三部分，是传入参数，就是数学函数中的自变量。

第四部分就是函数体，也就是要完成函数功能的逻辑代码，结果值是通过 return 语句进行返回的，而整个函数体的逻辑要包裹在一对大括号内部。

下面我们就来看一下在程序中如何使用函数功能：

 复制代码

```
1 #include <stdio.h>
2 int square(int x) { // 定义函数 square
3     return x * x;
4 }
5 int main() {
6     int n;
7     scanf("%d", &n);
8     printf("%d\n", square(n));
9     return 0;
10 }
```

上述代码中，在主函数中，我们读入一个整型数字 n，然后输出 n 的平方值。这里在计算 n 的平方值的时候，程序中调用了上面定义的 square 函数，那么 printf 函数相当于输出的是 square 函数的返回值，根据 square 函数的实现，如果传入的值是 x，那么返回值就是 $x * x$ ，即 x 的平方值。

这里需要你注意两个概念，我们将 n 传递给函数 square 的过程中，会涉及到 n 给 square 函数参数 x 赋值的过程。也就是说，主函数中的 n 变量和 square 函数参数 x 变量是两个相互独立的变量，其中 n 叫做“实参”，实际的参数，x 叫做“形参”，形式上的参数。

关于这个例子，我还要多说一句，还记得程序中的顺序结构吧，这是程序最基本的执行结构，也就是从左到右，从上到下的执行程序中的每一条语句。其实，函数和函数之间的关系，也可以理解为这种顺序执行的关系。


在这个例子中，我们在主函数中调用了 square 函数，也就意味着在这句话之前，程序中必须知道 square 函数的存在，因此 square 函数实现在了主函数之前。后面的文章中，你将

会学到，其实 square 函数不用实现在主函数之前也可以，这就要涉及到“声明”与“定义”的区别了，这个我后面再和你详细解释。

2. 普通变量的函数传递参数

了解了函数的基本知识以后，接下来让我们重点学习一下函数的参数传递过程，也就是上文中提到的“形参”和“实参”之间关系的问题。接下来的学习，我们都是围绕着一句话展开的，你先记住：**函数的参数传递过程，就是“实参”给“形参”赋值的过程，“实参”与“形参”之间互相独立，互不影响。**

下面先来看一下普通变量的传递过程，请看下面这段程序：

 复制代码

```
1 #include <stdio.h>
2 void add(int n, int m) {
3     n += m;
4     return ;
5 }
6 int main() {
7     int n, m;
8     scanf("%d%d", &n, &m);
9     add(n, m);
10    printf("%d\n", n);
11    return 0;
12 }
```

这段程序中，首先读入两个变量 n 和 m 的值，然后将 n 和 m 传递给一个名叫 add 的函数，add 函数的相关参数也叫 n 和 m，然后在 add 函数内部，将 m 累加到了 n 上面，之后函数返回结束，没有返回值。add 函数执行完后，回到主函数中，输出 n 的值。我的问题是，此时，n 的值有没有变化？

如果你实际运行这个程序，你会发现，n 的值不会改变，这就是我想让你记住的那句话，函数的参数传递过程，就是“实参”给“形参”赋值的过程。

这个程序中，主函数中的变量 n 就是“实参”，add 函数中的参数 n 就是“形参”，虽然两者名字一样，可完全是两个互相独立的变量。

两者有各自的存储空间，“实参”就是把自己存储空间中的值，复制一份给了“形参”，所以，在函数内部，我们实际修改的是“形参”中所存储的值，对主函数中的变量 n 毫无影响。整个过程如下图所示：

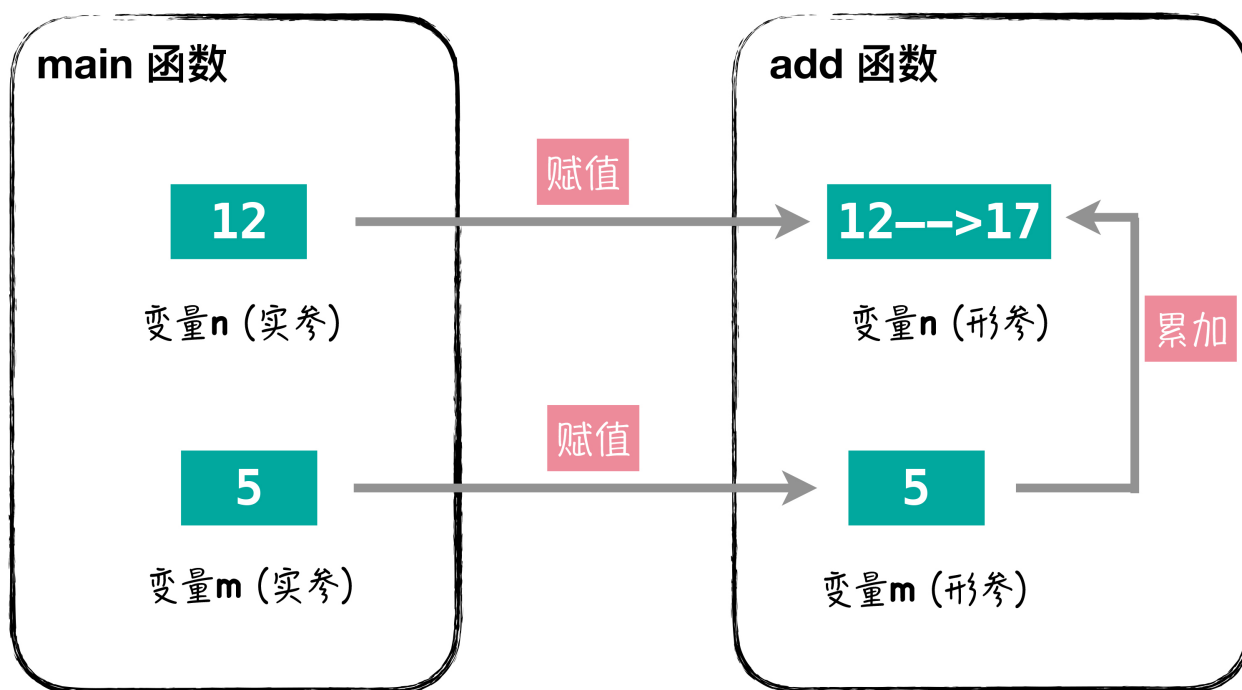


图2：实参、形参赋值示意图

如图所示，add 函数内部做的所有操作，都是在黄色的变量存储区内做的，对主函数中的变量存储区毫无影响。

那么如果我们想要改变 n 最后输出的值，你知道这个程序怎么改动呢？这里，你需要注意往下学习什么是传入参数和传出参数。

3. 数组的函数传参

看了普通变量的传参以后，下面来看一下数组作为参数时候的传参方式和特性，请看下面这段代码：

```
1 #include <stdio.h>
2 void add(int *p, int n) {
3     for (int i = 1; i < n; i++) {
4         p[0] += p[i];
5     }
6 }
```

[复制代码](#)

```
7     return ;
8 }
9 int main() {
10     int arr[10] = {1, 2, 3};
11     add(arr, 3);
12     printf("%d", arr[0]);
13     return 0;
14 }
```

这段程序中，主函数定义了一个拥有 10 个整型元素的数组，然后数组的前三位分别初始化为 1、2、3，之后将数组作为 add 函数的第一个参数，第二个参数是一个数字 3，add 函数的功能是将传入的数组中的前 n 个位置的值，累加到数组的第一个元素上。在 add 函数执行完后，在主函数中输出数组第一个元素的值。

对于这份代码的输出，你有什么预测么？在你做出预测之前，我提醒你注意一个地方，就是 add 函数中负责接收数组参数的第一个参数的类型，是一个指针类型，这里结合之前的知识就能理解了。数组名一般情况下代表了数组的首地址，将一个地址作为值传入函数，当然要用指针变量来进行接收了。

最后，你运行这段程序，会发现输出的结果是 6，意味着数组中的第一个元素的值发生了变化。再想想今天我们要记住的那句话：**函数的参数传递过程，就是“实参”给“形参”赋值的过程，“实参”与“形参”之间互相独立，互不影响。**

不是说互相独立么，怎么数组的第一个元素的值却改变了呢。没错，数组的第一个元素的值确实在函数内部被改变了，可这跟“实参”和“形参”的关系完全没有冲突。

请你注意，这里面我们的“实参”，实际上是数组的首地址，形参是存储这个首地址的函数参数中的那个指针变量。也就是说，在 add 函数内部，操作的地址空间，和主函数中的那个数组的空间是一个空间，这就是为什么传递数组时，相关元素的值在函数内部可以被改掉的一个原因，因为传递的是地址！

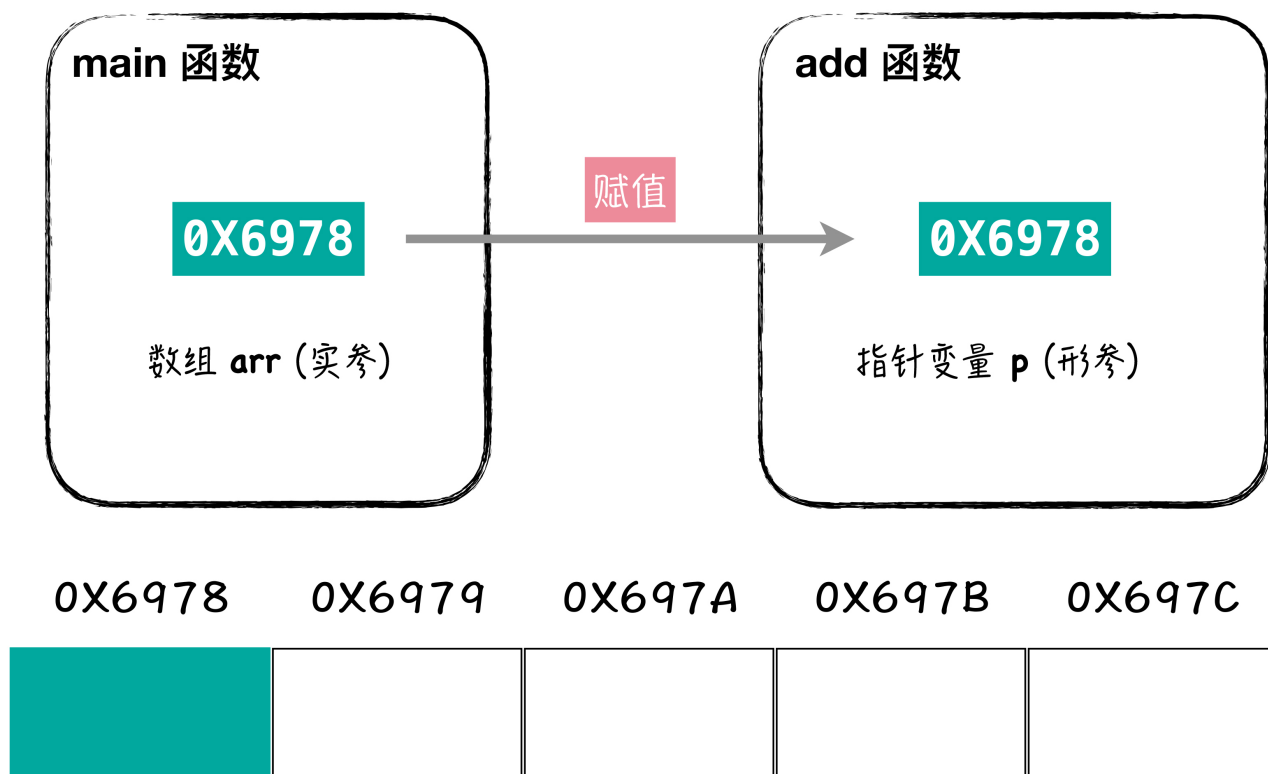


图3：数组传参过程


就如图 3 中所示，主函数中的数组 arr 将自己的首地址赋值给了指针变量 p，两者虽然互相独立，可只要不改变指针变量 p 中存储的地址，p[0] 和 arr[0] 实际上对应的就是同一片存储空间，所以修改 p[0] 的值，也相当于修改了 arr[0] 的值。

4. 传入与传出参数

学习了函数的基本知识以后，最后让我们来看两个逻辑上的概念“传入参数”与“传出参数”。

请看下面这段程序：

```
1 #include <stdio.h>
2 void calc(int x, int *p) {
3     *p = x * x;
4     return ;
5 }
6 int main() {
7     int n, m;
8     scanf("%d", &n);
9     calc(n, &m);
10    printf("%d\n", m);
```

 复制代码


```
11     return 0;
12 }
```

上面这段程序中，开始先定义了一个 calc 函数，calc 函数有两个参数，第一个是一个整型参数，第二个是一个整型地址，函数内部，将 x 的平方值存储到了 p 所指向的存储空间中。在主函数中调用了 calc 函数，分别传入 n 的值和 m 的地址，然后输出 m 的值，最后你会发现输出的 m 值，就是 n 的平方值。

在这里我们重点来讨论一下函数 calc 两个参数的本质作用。首先第一个参数 x，是外部传入的一个值，这个值在函数内部，要参与重要的运算过程，也就是说，这个值的感受更像是从外部传入到内部，然后在函数内部发挥作用，这种类型的参数，我们就叫“传入参数”。

而再看 calc 函数的第二个参数，传入的是一个地址。在函数内部的作用，只是将计算得到的一些结果存储在这个地址所指向的空间中，而记录的这些结果，在函数内部是没有用的，是要等到函数执行完后，回到调用者之后，例如上面的主函数内部，才有用。这一个参数的作用，更像是把值从 calc 内部带出到主函数内部而设计的，这类参数，我们叫做“传出参数”。

就像上面代码中看到的，“传入参数”一般就是把值传进去就行，而“传出参数”由于要把值从函数中带出来，一般要传变量地址进去，这样，函数内部才能准确的把结果写入到相关地址所对应的内存中。

一起动手，搞事情

我们又到了每天的“一起动手，搞事情”的环节，今天呢，将给你留两个思考题。

思考题（1）：数组和函数的思考

请思考如下两个概念的异同：

1. 一个整型数组元素，例如：arr[100]
2. 一个传入整型并且返回整型的函数，例如：func(100)

这是一个开放思考题，写出你的理解及思考过程即可。

思考题（2）：如何确定存在知识的盲区

什么叫“存在知识的盲区”呢？就是当你面对一片黑暗的时候，你可以确定这里一定有知识，而不仅仅只是一片黑暗。就像今天我们学习了函数的相关知识，自然的，就会反问自己一句，这些就是函数知识的全部了么？我们如何来确定这个问题的答案呢？很简单，根据已知推未知。

我们假设现在学习的内容，已经是函数知识的全部了，也就是说，只要是函数，我们就能用我们现有知识对其加以解释。

那么，在之前，我们已知的函数中，有两个很基础，也很重要函数，一个是 scanf 函数，一个是 printf 函数。

随便来看一个，例如来看 scanf 函数，当我问你，scanf 函数，传入几个参数的时候，你会发现是若干个。第一个参数是一个字符串，往后的参数，是根据字符串中格式占位符的数量而定的。在不要求你实现 scanf 函数功能的情况下，你能将 scanf 函数包含参数定义的形式写出来么？直到这里，我们就发现了一个存在知识的盲区。


所以，没有知识的盲区，只是盲区，发现有价值盲区的能力，也是我们要锻炼的重要能力。既然发现了这个知识盲区，给你留个小作业，自学“可变参函数”相关的知识吧。

实现 my_scanf 函数

准备完了对于函数的基础知识以后，再回到今天一开始提到的任务。首先来分析一下只转换一个整型参数的 my_scanf 函数应该如何进行实现。

第一步，我们先来看参数设计，第一个参数，应该是一个字符串类型的“传入参数”，代表要转换成整型信息的字符串信息。第二个参数，应该是一个指针类型的“传出参数”，指向存储转换结果的内存区域。

具体功能实现，请看下面这段代码：

 复制代码

```
1 #include <stdio.h>
2 void my_scanf(char *str, int *ret) {
3     int num = 0, flag = 0;
4     if (str[0] == '-') str += 1, flag = 1;
```

```

5     for (int i = 0; str[i]; i++) {
6         num = num * 10 + (str[i] - '0');
7     }
8     if (flag == 1) num = -num;
9     *ret = num;
10    return ;
11 }
12 int main() {
13     char str[1000];
14     int n = 65;
15     scanf("%s", str);
16     my_scanf(str, &n);
17     printf("n = %d\n", n);
18     return 0;
19 }

```

这段代码中，实现了 my_scanf 函数。在看 my_scanf 函数具体逻辑之前，先来看一下主函数里面都写了些什么。

主函数的头两行定义了两个变量，一个是字符数组 str，另外是一个整型变量 n，然后读入一个字符串，将其保存在字符数组中。再之后，使用 my_scanf 函数将字符数组中的字符串信息，转换为整型信息存储在 n 中，最后，使用 printf 函数输出 n 的值，加以确认。

看完了主函数以后，再来看一下 my_scanf 函数的具体实现。my_scanf 函数第一行定义了两个变量，一个用于存放转换结果的 num 变量，另一个 flag 变量用来标记正负数的，0 代表正数，1 代表负数。

第 2 行判断字符串中的第一位是不是字符 '-'，如果是字符 '-'，就将 flag 标记为 1，并且把 str 字符指针所指的位置，向后跳动一位，因为 '-' 后面就是要转换的第一个数字字符了。之后遍历字符串剩余的每一位，每次将当前字符所代表的数字，放到 num 数字的末尾。

其中 str[i] - '0'，就是将相关的数字字符，转换成对应的数字。之前我们说了，任何一个信息在底层存储的时候，都是二进制信息表示，也就是说，都可以转换为一个十进制数字，字符信息也不例外。其中字符 '0' 所对应的底层数字是 48，字符 '1' 是 49，字符 '2' 是 50，依次类推。所以当我们用 '2' - '0' 的时候，相当于 50 - 48，得到的结果就是数字 2。

最后把 num 中的值拷贝到 ret 所指向的存储区中，也就是主函数中的 n 变量的内存区中。至此我们就完成了一个整型参数的 my_scanf 函数的实现。接下来，运用“可变参函数”的相关知识，改写这个程序，去独立完成最终形态的程序吧。

课程小结

今天讲的内容呢，是里程碑式的一课，到目前为止，你已经学会了将程序模块化的最基本技术：函数。也是从这一课开始，后面我将越来越多的起到引导你的作用，逐渐帮你撤掉学习中对我的依赖，如果后续学习中遇到什么问题，咱们随时在留言区中讨论。

最后呢，我来给你总结一下今天课程的重点，只希望你记住三点：

1. 函数的作用，是做功能封装，以便在程序其他地方复用相关功能。
2. C 语言中的函数的传参过程，是实参给形参赋值的过程，改变形参的值，不会影响实参。
3. 在函数参数设计中，一定要分清楚，传入参数和传出参数在功能上的差别。

好了，今天就到这里了，我是胡光，我们下次见。



人人都能学会的 编程入门课

>>> 每天 10 分钟，轻松学编程

胡光

原百度高级算法研发工程师



新版升级：点击「 请朋友读」，20位好友免费读，邀请订阅更有**现金**奖励。

上一篇 08 | 指针系列（二）：记住，指针变量也是变量

下一篇 10 | 预处理命令（上）：必须掌握的“黑魔法”，让编译器帮你写代码

精选留言 (2)

写留言



徐洲更

2020-01-25

知识点:

1. 函数的实参和形参，形参修改不会修改实参
2. 为了在函数中更改函数外的参数，可以传入指针，直接修改内存里的内容
3. 快速将'0'-'9'字符转成数字0-9的方法, '9' - '0'
4. 流处理方式转换字符串为数字, `num = num * 10 + (str[i] - '0')`...

展开

作者回复: 不错！作业题目做的很棒。完全达到了自学的要求！



1



小林coding

2020-02-01

老师，我的可变参数的myScanf函数

代码：

```
#include <iostream>
```

```
#include <stdarg.h>
```

...

展开

作者回复: 可以打80分，稍微有点儿缺憾的就是：没有考虑负数的情况哦。d(^_^o)

1

