**C语言概述**

**C语言的一个简单实例：**

#include <stdio.h>

int main(){

int num;

num=1;

printf("I am a simple"); /\*输出I am a simple\*/

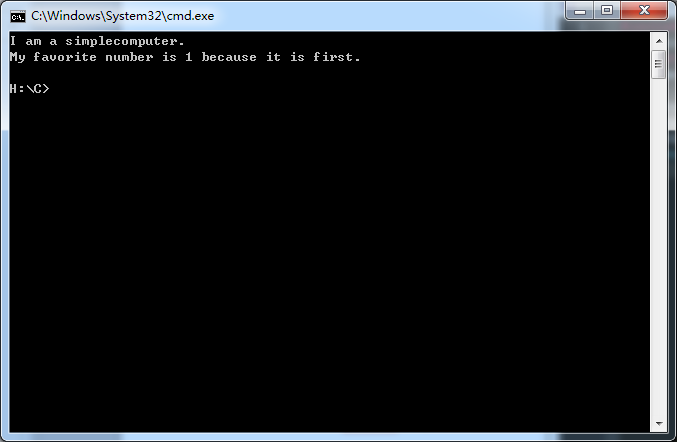
printf("computer.\n"); /\*输出computer\*/

printf("My favorite number is %d because it is first.\n",num); /\*输出My favorite number is 1 because it is first.\*/

return 0;

}

运行结果：



代码解释：

#include <stdio.h> 包含另一个文件

Int main() 函数名

/\*这是一个注释\*/ 注释

{ 函数体的开始

Int num 声明变量

Num =1; 赋值语句

Printf(“I am a simple”); 一个函数调用语句

Return 0; 返回语句

} 函数体的结束

**程序细节：**

# include <stdio.h> 该语句引用了文件stdio.h的完整内容。实际上，它是一种剪切和粘贴操作，这样可以方便地在多个程序间共享公用的信息，include语句是C预处理器的指令

预处理：C编译前对源代码做一些准备工作；这称为预处理

Stdio.h文件作为所有C编译包的一部分提供，它包含了有关输入和输出函数（例如：printf（））的信息以供编译器使用。这个名字代表标准输出和标准输入头文件，在C世界中人们称出现在文件顶部的信息集合为头，C实现常常有很多头文件。最重要的是头文件包括了建立最终的可执行程序编译时所需要用到的信息。例如，他们可以定义常量，或者说明函数名以及该函数如何使用。但函数的实际代码被包含在一个预编译器，把您要的程序正确的组合在一起

PS：C预处理语句并不是C语言的语句

**Main()函数：**

Main()函数是C的主体函数，C执行代码的时候从main()函数开始执行，main()必须是开始的函数。Int指明了main()函数返回的类型。就意味着main()函数返回的结果是整数，main()函数是C语言的基本模块

**声明：**

在C里面每使用一个变量的时候都需要先进行声明，例如：

Int demo; /\*声明一个int类型的demo变量\*/

Int a,b,c,d; /\*一次声明多个变量\*/

声明的规则：数据类型 变量名;

在main()函数里面声明的变量为全局变量，在其他函数里面的声明的变量只能是局部变量。全局变量可以在任何地方访问，局部变量只能在哪个函数里面访问这个变量

**赋值：**

赋值，赋值语句是C的最基本操作之一。它用于将一个值赋予给一个变量

例如:

Int num;

Num=1; or int num=1;

**Printf()函数：**

C语言的标准输出函数，用于输出字符串或字符或变量的值

**Return语句:**

Return语句是返回语句是程序的最后一个语句，在int main()函数里返回的是一个整数。C语言标准要求mian()函数这么做。带有返回值的C语言函数要使用一个return语句，该语句包括关键字return,后面紧跟着要返回的值，然后是一个分号。对于main()函数来说少了return语句就会给你个警告但并不影响程序执行

**更进一步：**

#include <stdio.h>

int main(){

int feet,fathoms;

fathoms=2;

feet=6\*fathoms;

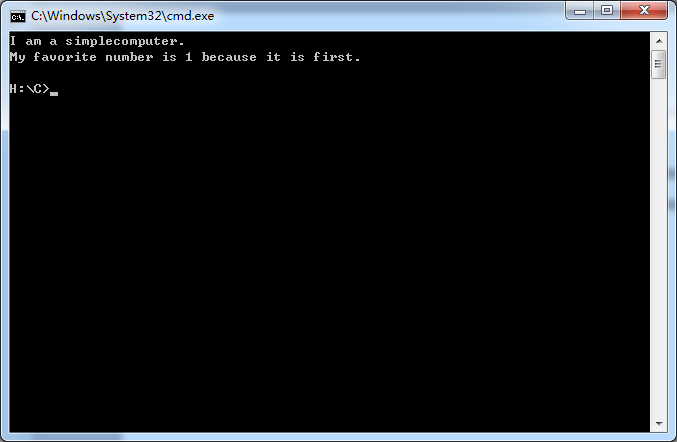
printf("There are %d feet in %d fathoms!\n",feet,fathoms);

printf("Yes,I said %d feet!\n",6\*fathoms);

return 0;

}

运行结果：



**多个函数：**

#include <stdio.h>

void bulter();

int main(){

printf("I will summon the bulter function.\n");

bulter();

printf("Yes,Bring me some tea and writeable CD-ROMS.\n");

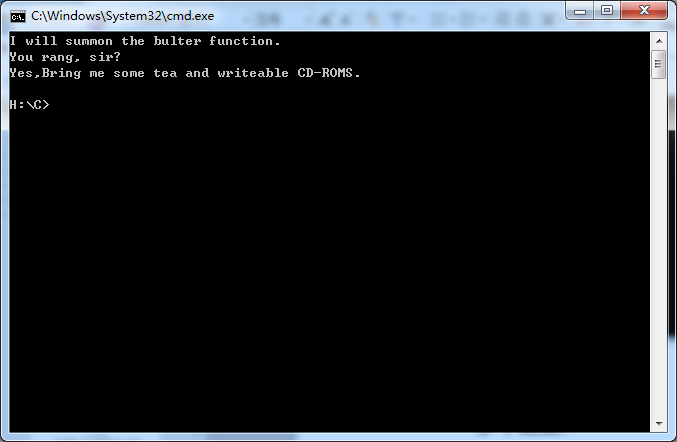
return 0;

}

void bulter(){

printf("You rang, sir?\n");

}



PS：使用函数的时候也需要和变量一样先声明

例如：

Void demo();

规则：数据类型 函数名(是否有值)

**变量覆盖例子：**

#include <stdio.h>

int main(){

int a,b,c;

a=10;

b=a; /\*变量b等于变量a的值\*/

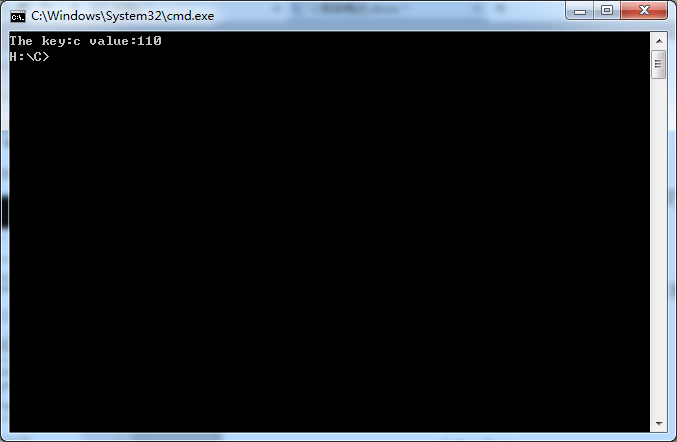
c=a+100; /\*变量c等于变量a的值+100\*/

printf("The key:c value:%d",c);

return 0;

}

运行结果：



**关键字与保留标识符：**

像int,if这些关键字，是不能定义为变量的。否则的话可能会引发问题，保留标识符包括那些以下划线字符开始的标识符和标准函库函数的名字，例如printf()

C语言关键字列表：

|  |  |  |  |
| --- | --- | --- | --- |
| auto | enum | restrict | unsigned |
| break | extern | return | void |
| case | float | short | volatile |
| char | for | signed | while |
| const | goto | sizeof | \_Bool |
| continue | if | static | \_Complex |
| default | inline | struct | \_Imaginary |
| do | int | switch |  |
| double | long | typedef |  |
| else | register | union |  |

**语法错误和语义错误：**

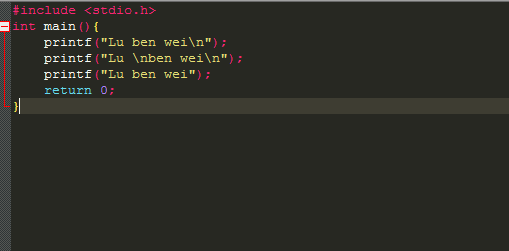
语法错误是代码不规范或者写错了的叫语法错误，语义错误是指，含义上的错误

**总结：**

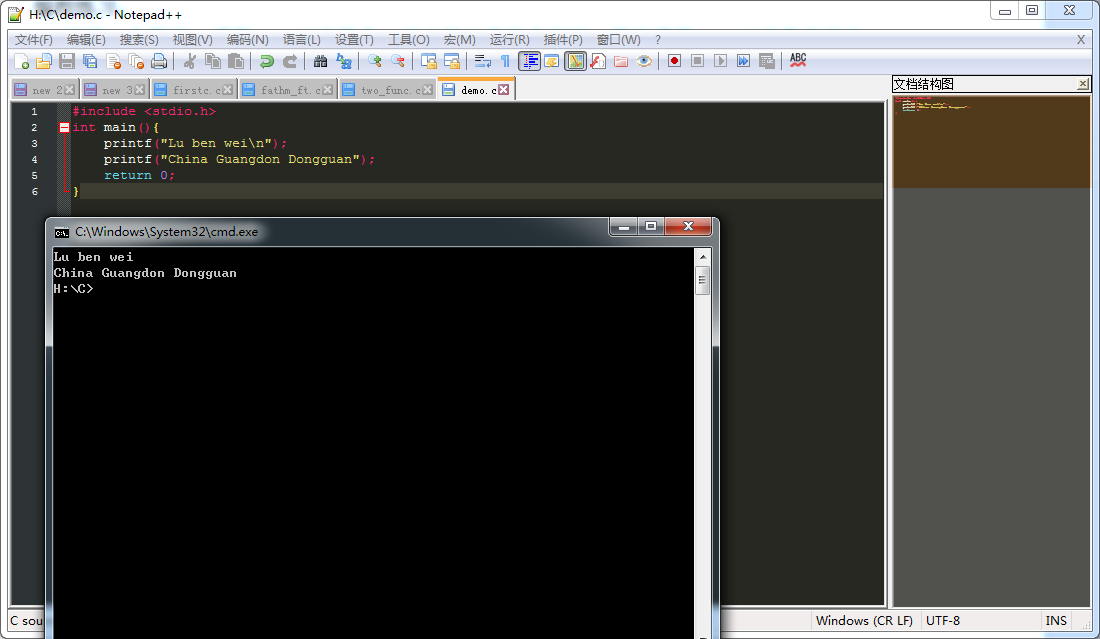
C语言程序是由一个或者多个函数组成的。每一个C程序都必须包含一个名为main()的函数，因为当程序开始时要调用该函数。一个简单的函数结构如下：函数头后面跟着一个开始花括号，后面是构成函数体的语句，然后是终止作用的结束话括号

**编程练习：**

1. 编写一个程序，调用printf()函数在一行上输出您的名和姓，再次调用printf()函数在两个单独的行上输出您的名和姓



1. 编写一个程序输出您的姓名和地址



1. 编写一个程序，把您的年龄转换成天数并显示二者的值。不用考虑平年和闰年的问题

#include <stdio.h>

int main(){

int age=0;

printf("Input your age:");

scanf("%d",&age);

if(age<=0||age>=200){

printf("Your age is incorrect,please input again");

scanf("%d",&age);

}

printf("Your age is:%d,and the days are:%d\n",age,age\*365+age/4);

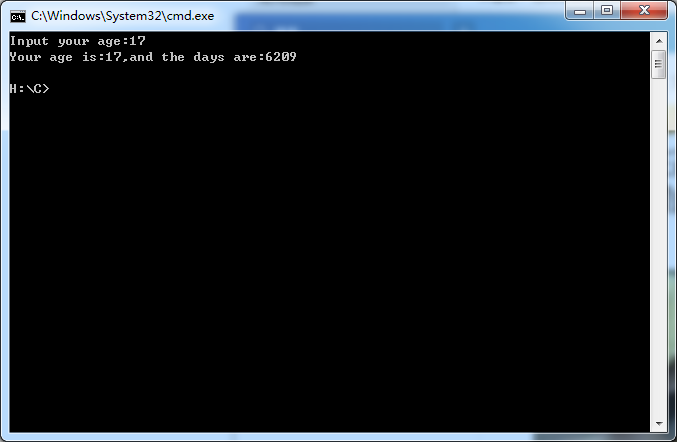
return 0;

}

Scanf() /\*让用户输入值然后赋值到变量上\*/

&是取地址符

运行结果：



1. 编写一个能产生下面输出的程序：

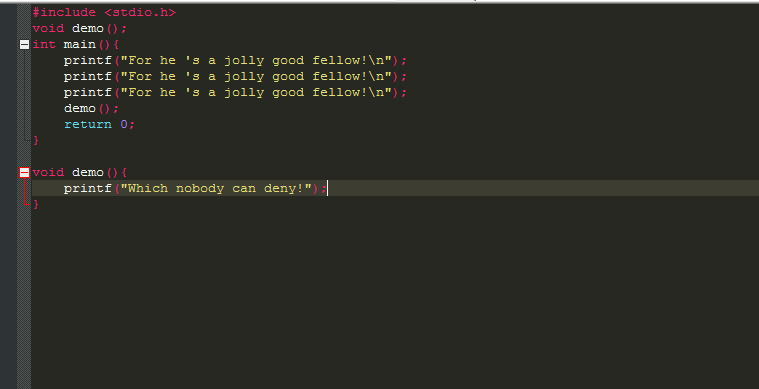
For he’s a jolly good fellow!

For he’s a jolly good fellow!

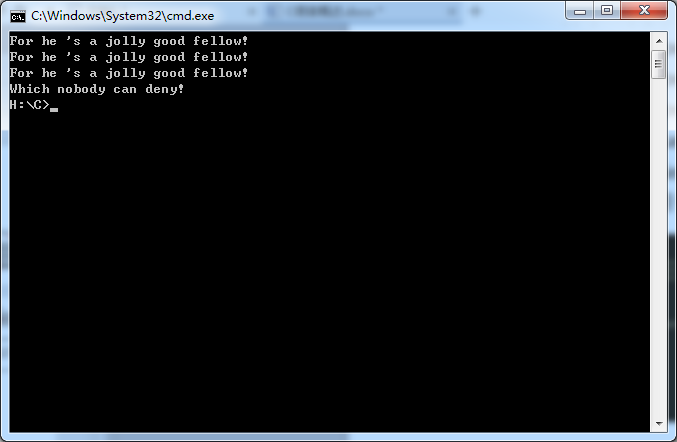
For he’s a jolly good fellow!

Which nobody can deny!

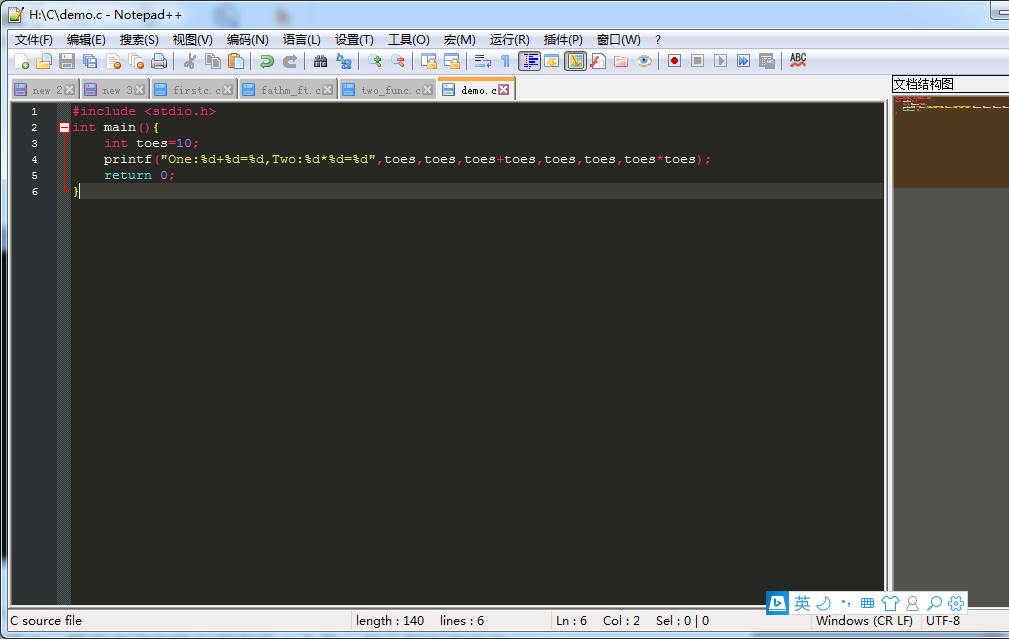
程序中除了main()函数之外，要使用两个用户定义的函数：一个用于把上面的夸奖信息输出一次；另一个用于把最后一行输出一次



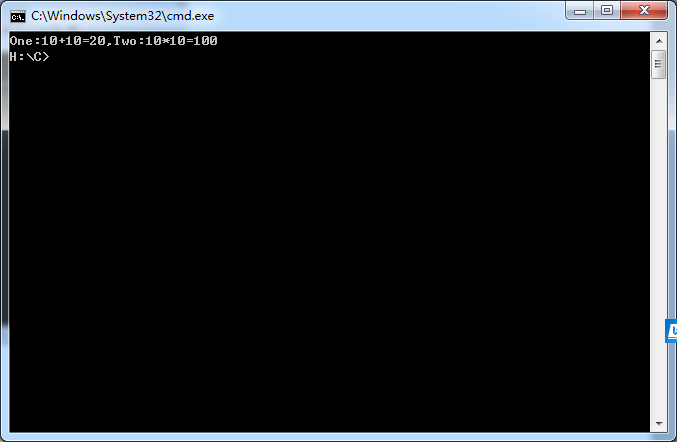
运行结果：



1. 编写一个程序，创建一个名为toes的整数变量。让程序把toes设置为10。再让程序计算两个toes的和以及toes的平方。程序应该能输出所有的3个值，并分别标识它们。



运行结果：



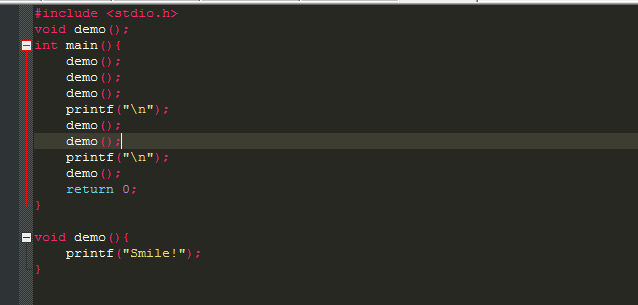
1. 编写一个能产生下列输出的程序

Smile!Smile!Smile!

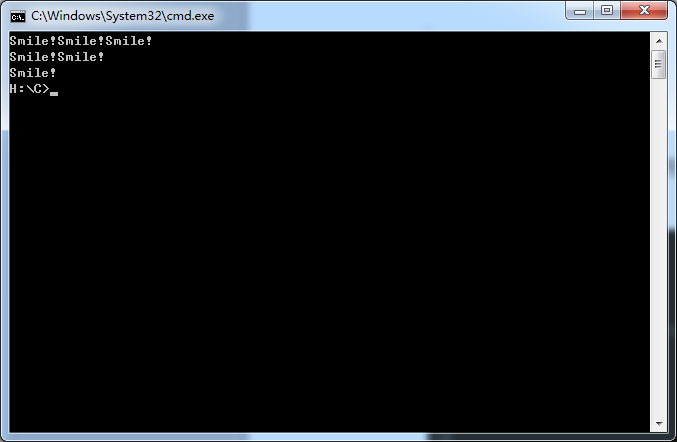
Smile!Smile!

Smile!

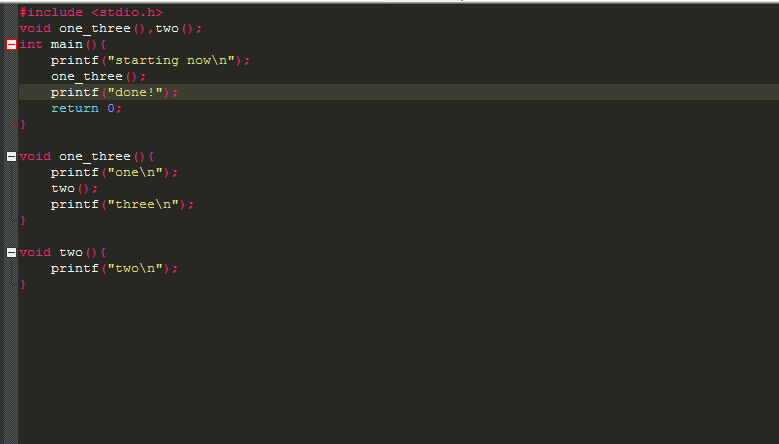
在程序定义一个能显示字符串smile!一次的函数，并在需要的时候使用该函数



运行结果：



1. 编写一个程序，程序中要调用名为one\_three()的函数。该函数要在一行中显示的单词“one”,再调用two()函数，然后再在另一行中显示单词“three”。函数two()应该能在一行中显示单词“two”。Main()函数应该在调用one\_three()函数之前显示短语“starting now”,函数调用之后还要显示“done!”。



运行结果：

