

# § . 基础知识题 - C方式输入输出的格式化控制



要求:

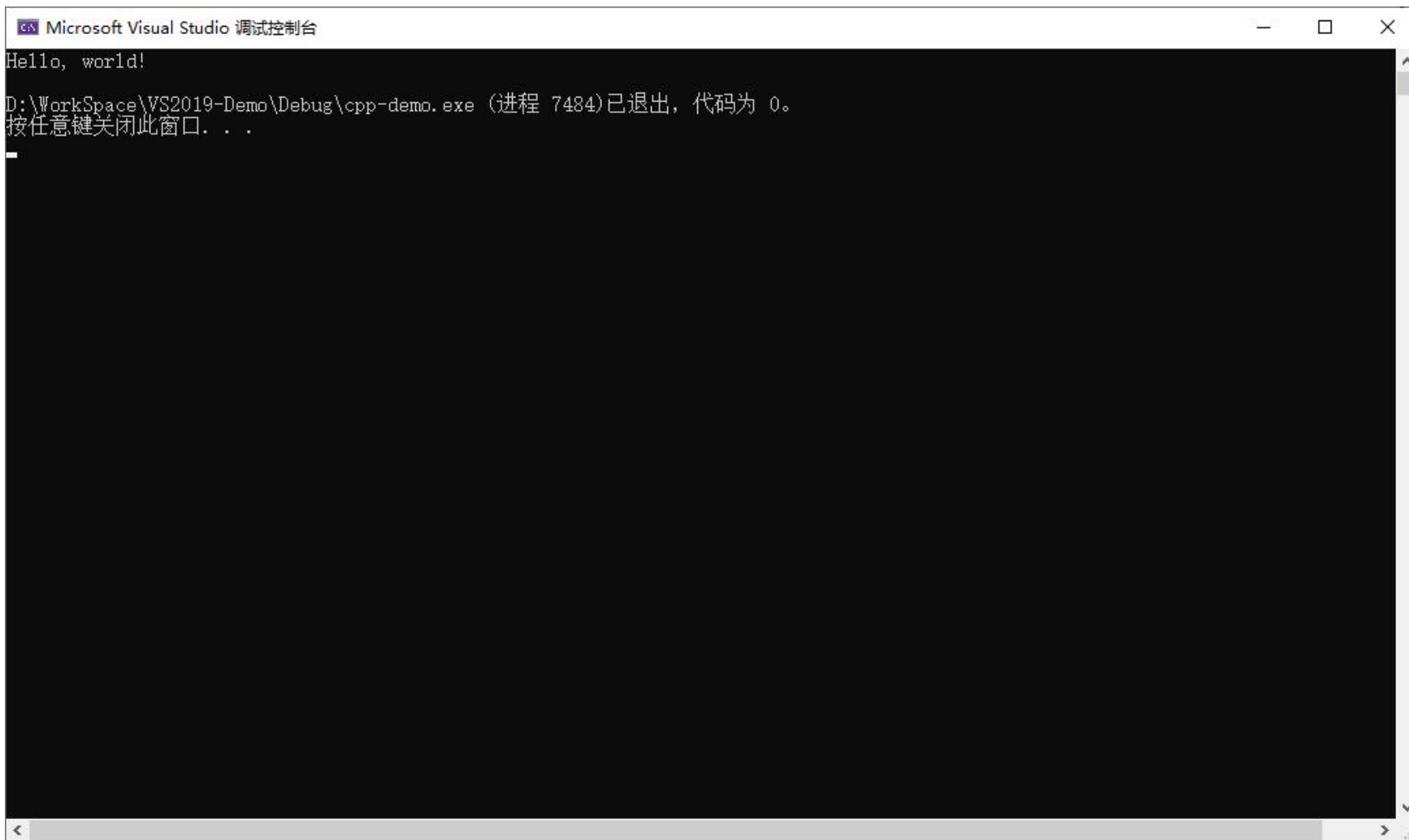
- 1、完成本文档中所有的题目并写出分析、运行结果
- 2、无特殊说明，均使用VS2022编译即可
- 3、直接在本文件上作答，**写出答案/截图（不允许手写、手写拍照截图）**即可；填写答案时，为适应所填内容或贴图，**允许调整**页面的字体大小、颜色、文本框的位置等
  - ★ 贴图要有效部分即可，不需要全部内容
  - ★ 在保证一页一题的前提下，具体页面布局可以自行发挥，简单易读即可
  - ★ **不允许**手写在纸上，再拍照贴图
  - ★ **允许**在各种软件工具上完成（不含手写），再截图贴图
  - ★ 如果某题要求VS+Dev的，则如果两个编译器运行结果一致，贴VS的一张图即可，如果不一致，则两个图都要贴
- 4、转换为pdf后提交
- 5、**9月26日前**网上提交本次作业（在“文档作业”中提交）

## §. 基础知识题 - C方式输入输出的格式化控制

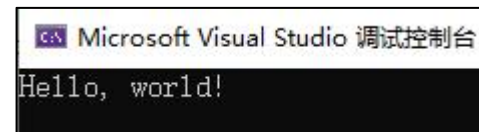


贴图要求：只需要截取输出窗口中的有效部分即可，如果全部截取/截取过大，则视为无效贴图

例：无效贴图

A screenshot of the Microsoft Visual Studio debug console window. The window is titled "Microsoft Visual Studio 调试控制台". It contains the following text: "Hello, world!", "D:\Workspace\VS2019-Demo\Debug\cpp-demo.exe (进程 7484)已退出, 代码为 0.", and "按任意键关闭此窗口. . .". The window is large and shows the full output of the program.

例：有效贴图

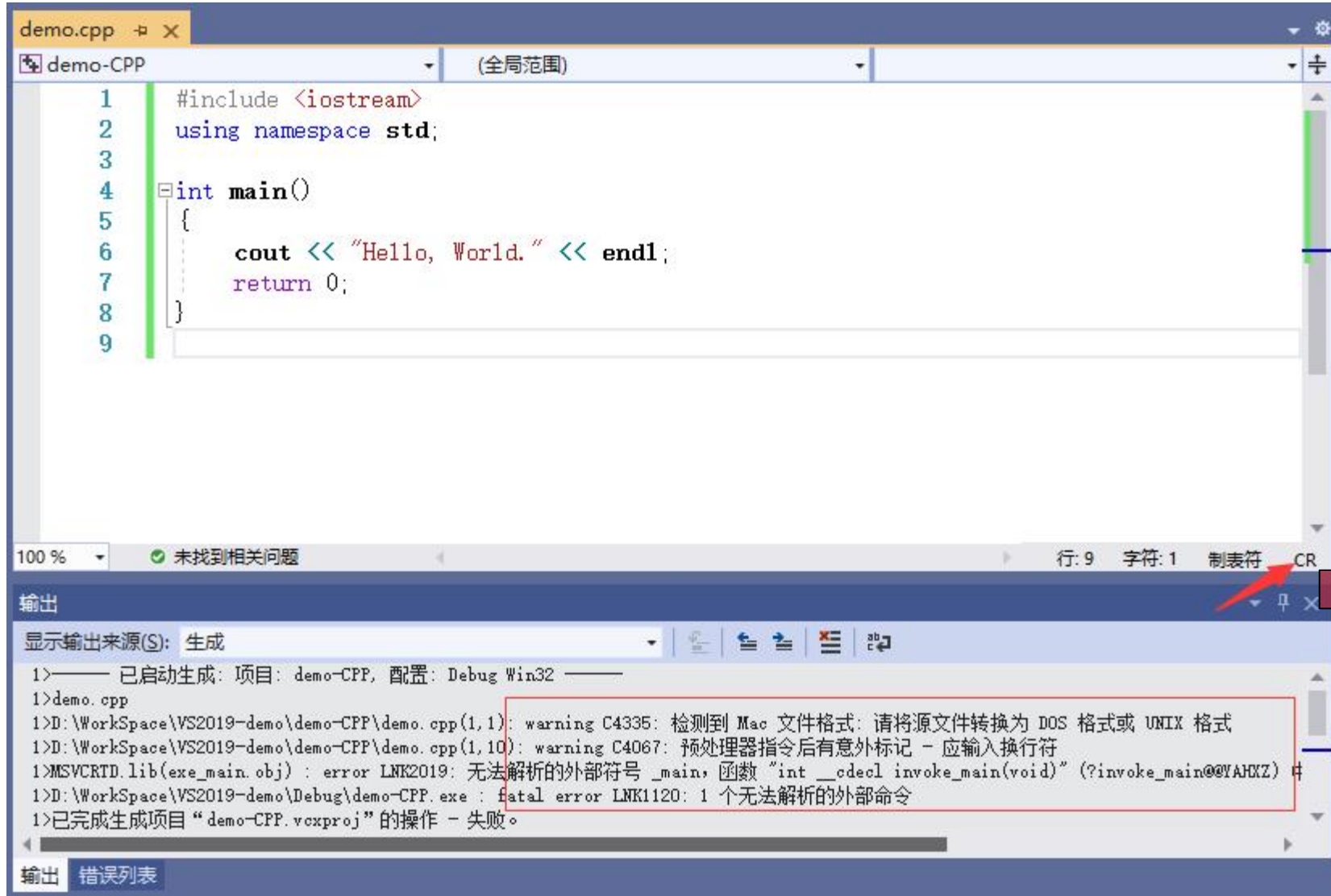
A screenshot of the Microsoft Visual Studio debug console window, cropped to show only the "Hello, world!" output. The window title is "Microsoft Visual Studio 调试控制台".



## §. 基础知识题 - C方式输入输出的格式化控制

附：用WPS等其他第三方软件打开PPT，将代码复制到VS2022中后，如果出现类似下面的**编译报错**，则观察源程序编辑窗

的右下角是否为CR，如果是，单击CR，在弹出中选择CRLF，再次CTRL+F5运行即可



## §. 基础知识题 - C方式输入输出的格式化控制



特别提示:

- 1、做题过程中, 先按要求输入, 如果想替换数据, 也要先做完指定输入
- 2、如果替换数据后出现某些问题, 先记录下来, 不要问, 等全部完成后, 还想不通再问 (也许你的问题在后面的题目中有答案)
- 3、不要偷懒、不要自以为是的脑补结论!!!
- 4、先得到题目要求的小结论, 再综合考虑上下题目间关系, 得到综合结论
- 5、这些结论, 是让你记住的, 不是让你完成作业后就忘掉了
- 6、换位思考(从老师角度出发), 这些题的目的是希望掌握什么学习方法?

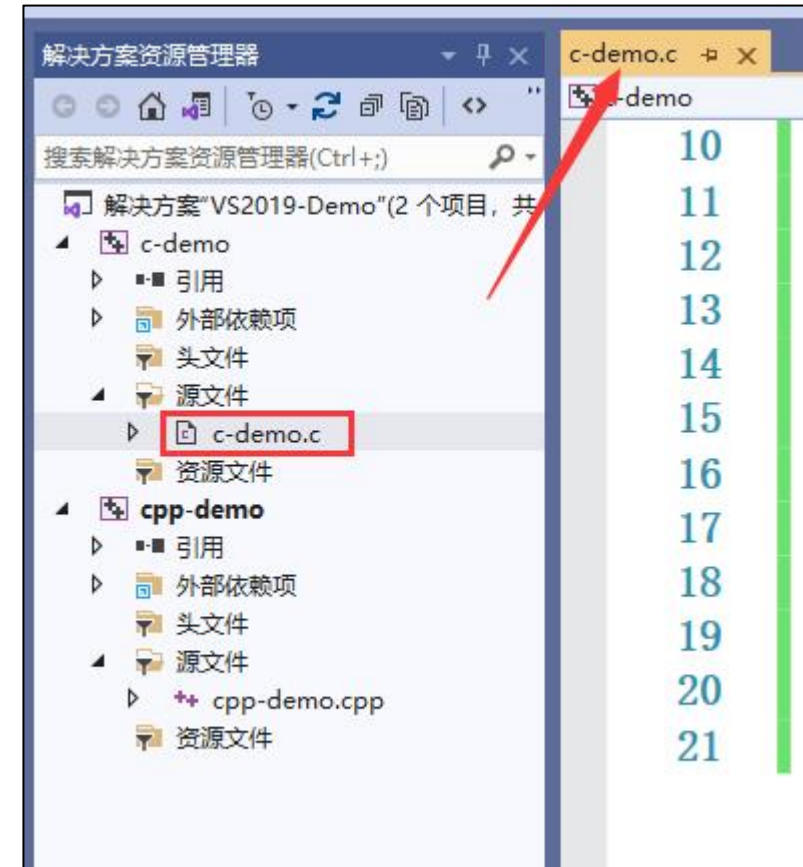
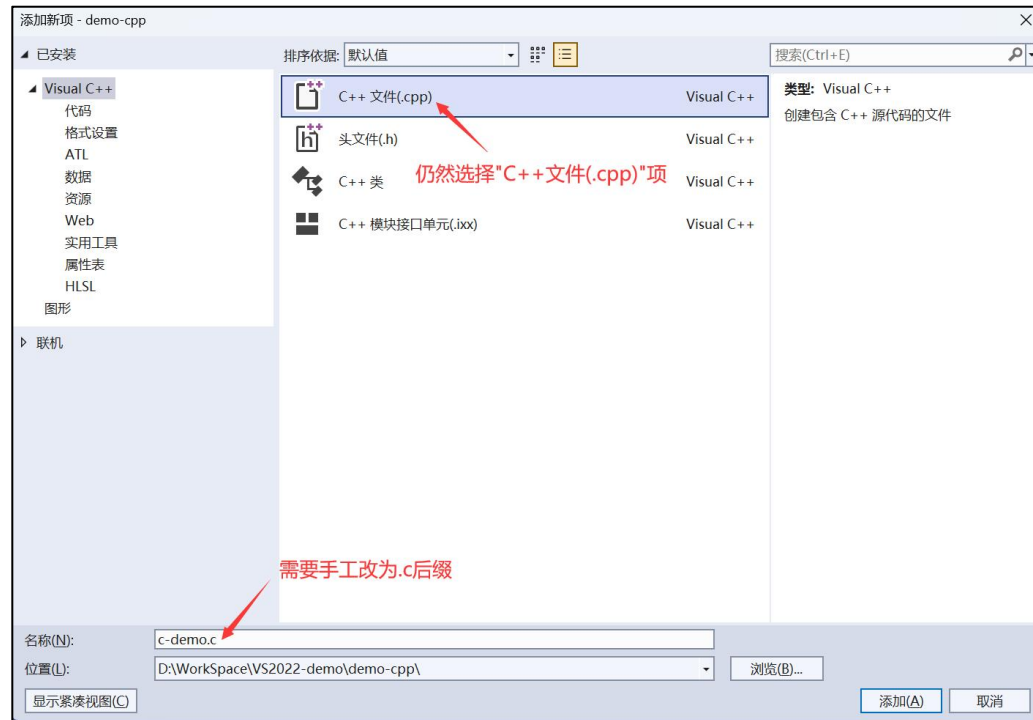


## § . 基础知识题 - C方式输入输出的格式化控制

本次作业特别要求:

1、建立解决方案-项目-源程序文件时,一定要.c后缀,不要.cpp后缀!!!

**提醒:** .c和.cpp的报错表现不同,按.cpp做会影响分数



2、如果是warning+有结果,则warning+运行结果两者的截图都要!!!



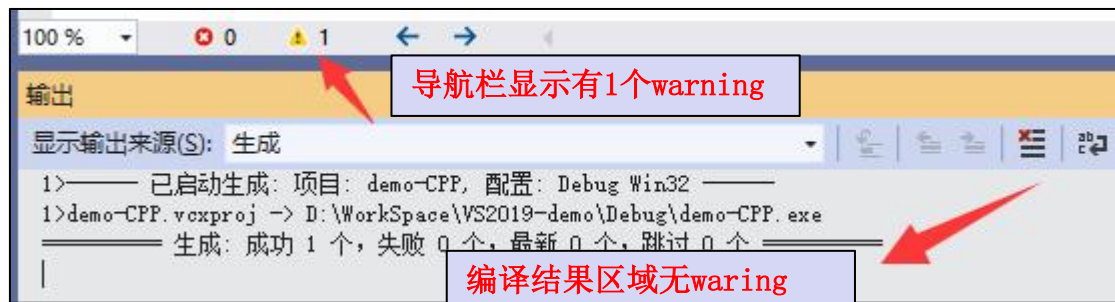
# §. 基础知识题 - C方式输入输出的格式化控制



★ 关于VS2022在C/C++中使用scanf时，报warning的统一处理方法(更多内容，参考编号为030105的附件文档及视频)

```
demo.cpp - demo-CPP
1  #define _CRT_SECURE_NO_WARNINGS
2  #include <stdio.h>
3  int main()
4  {
5      int k;
6      scanf("%d", &k);
7      printf("%d\n", k);
8      return 0;
9  }
```

```
demo.cpp - demo-CPP
1  #define _CRT_SECURE_NO_WARNINGS
2  #include <iostream>
3  using namespace std;
4  int main()
5  {
6      int k;
7      scanf("%d", &k);
8      printf("%d\n", k);
9      return 0;
10 }
```



- 1、如上图两个程序，按 CTRL+F5 可以正确运行，编译结果显示区域未出现warning，但导航栏提示有一个warning
- 2、点开导航栏后出现一个warning信息
- 3、这属于VS智能提示（IntelliSense）的警告，这种级别的警告暂时忽略，不需要消除，也不计入会扣分的warning的计数项





# § . 基础知识题 - C方式输入输出的格式化控制

## 1. 格式化输出函数printf的基本理解

形式：printf(格式控制表列, 输出表列);

格式控制表列的内容:

格式说明：以%开始+格式字符, 表示按格式输出

普通字符(含转义符)：原样输出

输出表列:

要输出的数据（常量、变量、表达式、函数）

常用的格式符种类:

printf所用的格式字符的种类:

d, i	带符号的十进制形式整数(正数不带+)
o	八进制无符号形式输出整数(不带前导0)
x, X	十六进制无符号形式输出整数(不带前导0x)
u	十进制无符号形式输出整数
c	以字符形式输出(一个字符)
s	输出字符串
f	以小数形式输出浮点数
e, E	以指数形式输出浮点数
g, G	从f, e中选择宽度较短的形式输出浮点数

printf所用的附加格式字符的种类:

字母l	表示长整型整数, 用于d, o, x, u前
字母h	表示短整型整数, 用于d, o, x, u前
正整数m	表示输出数据的宽度
正整数.n	对浮点数, 表示n位小数 对字符串, 表示前n个字符
-	输出左对齐

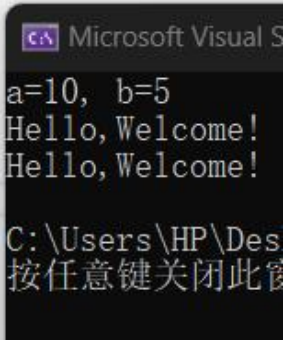
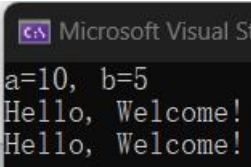
本页不用作答



# § . 基础知识题 - C方式输入输出的格式化控制

## 1. 格式化输出函数printf的基本理解

A. 观察下列程序的运行结果，回答问题并将程序的运行结果截图贴上(如果有错则贴错误信息截图)

<pre>#include &lt;stdio.h&gt;  int main() {     int a=10, b=5;     printf("a=%d, b=%d\n", a, b);      printf("Hello, Welcome!\n");     printf("Hello, Welcome\x21\n");     return 0; }</pre>	<p>//写出与左侧程序输出完全一致的，用C++方式的cout实现的代码 //贴源码或截图均可</p> <pre>#include&lt;iostream&gt; using namespace std;  int main() {     int a = 10, b = 5;     cout &lt;&lt; "a=" &lt;&lt; a &lt;&lt; ", " &lt;&lt; " " &lt;&lt; "b=" &lt;&lt; b &lt;&lt; endl;      cout &lt;&lt; "Hello, Welcome!\n";     cout &lt;&lt; "Hello, Welcome\x21\n";     return 0; }</pre> 
<p>运行结果:</p> 	
<p>\x21是哪个ASCII字符的16进制转义表示? <u>是!</u>的转义表示 转义符在格式控制表列中的输出形式 是: <u>字符</u> (字符/整数/转义符)</p>	


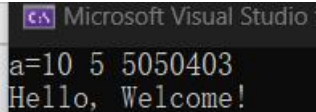




# § . 基础知识题 - C方式输入输出的格式化控制

## 1. 格式化输出函数printf的基本理解

B. 观察下列程序的运行结果，回答问题并将程序的运行结果截图贴上(如果有错则贴错误信息截图)

<pre>#include &lt;stdio.h&gt;  int main() {     int a=10, b=5;     printf("a=%d\n", a, b);      printf("Hello, Welcome!\n");     return 0; }</pre>	<pre>#include &lt;stdio.h&gt;  int main() {     int a=10, b=5;     printf("a=%d %d %d\n", a, b);      printf("Hello, Welcome!\n");     return 0; }</pre>
<p>运行结果:</p>  <p>Users\HP\Desktop\c\Project1\111.c(7,25): warning C4474: printf: 格式字符串中传递的参数太多 C:\Users\HP\Desktop\c\Project1\111.c(7,25): 占位符和其参数预计 1 可变参数，但提供的却是 2 参数</p>	<p>运行结果:</p>  <p>1&gt;C:\Users\HP\Desktop\c\Project1\111.c(6,12): warning C4473: "printf": 没有为格式字符串传递足够的参数 1&gt; C:\Users\HP\Desktop\c\Project1\111.c(6,12): 1&gt; 占位符和其参数预计 3 可变参数，但提供的却是 2 参数 1&gt; C:\Users\HP\Desktop\c\Project1\111.c(6,12): 1&gt; 缺失的可变参数 3 为格式字符串 "%d" 所需</p>
<p>结论: 如果%d(格式符的数量) <b>小于</b>后面输出表列的数量, 则 <u>会报warning, 运行结果只会输出a</u></p>	<p>结论: 如果%d(格式符的数量) <b>大于</b>后面输出表列的数量, 则 <u>会报warning, 运行结果会出现三个, 但第三个值为不可信值</u></p>



## §. 基础知识题 - C方式输入输出的格式化控制

### 1. 格式化输出函数printf的基本理解

C. 观察下列程序的运行结果，回答问题并将程序的运行结果截图贴上(如果有错则贴错误信息截图)

```
#include <stdio.h>

int main()
{
    int a=10, b=5;
    int ret1, ret2, ret3, ret4, ret5;

    ret1 = printf("a=%d, b=%d\n", a, b);
    ret2 = printf("a=%d b=%d\n", a, b); //跟上面比，少一个逗号

    ret3 = printf("a=%d\n", a*1000);

    ret4 = printf("Hello\n");
    ret5 = printf("Hello"); //跟上面比，少一个\n
    printf("\n");

    printf("%d %d %d %d %d\n", ret1, ret2, ret3, ret4, ret5);

    return 0;
}
```

运行结果:

```
Microsoft Visual Studio 调试
a=10, b=5
a=10 b=5
a=10000
Hello
Hello
10 9 8 6 5
```

对运行结果进行分析后，你认为printf的返回值的含义是：

printf返回值的含义是输出字符串的字符数（包括换行符，\n算一个字符）  
eg. ret2的分析：a = 1 0 空格 b = 5 \n 共9个字符，故返回值为9。（其余同理）



# §. 基础知识题 - C方式输入输出的格式化控制

## 1. 格式化输出函数printf的基本理解

D. 观察下列程序的运行结果，回答问题并将程序的运行结果截图贴上(如果有错则贴错误信息截图)

```
#include <stdio.h>

int main()
{
    short a = -2;
    printf("a=%hi %hd %hu %ho %hx %hX\n", a, a, a, a, a, a);
    printf("a=%i %d %u %o %x %X\n", a, a, a, a, a, a);
    printf("a=%li %ld %lu %lo %lx %lX\n", a, a, a, a, a, a);

    unsigned short b = 40000;
    printf("b=%hi %hd %hu %ho %hx %hX\n", b, b, b, b, b, b);
    printf("b=%i %d %u %o %x %X\n", b, b, b, b, b, b);
    printf("b=%li %ld %lu %lo %lx %lX\n", b, b, b, b, b, b);

    int c = 70000;
    printf("c=%hi %hd %hu %ho %hx %hX\n", c, c, c, c, c, c);
    printf("c=%i %d %u %o %x %X\n", c, c, c, c, c, c);
    printf("c=%li %ld %lu %lo %lx %lX\n", c, c, c, c, c, c);

    return 0;
}
```

运行结果:

```
Microsoft Visual Studio 调试控制台
a=-2 -2 65534 177776 fffe FFFE
a=-2 -2 4294967294 3777777776 ffffffff FFFFFFFF
a=-2 -2 4294967294 3777777776 ffffffff FFFFFFFF
b=-25536 -25536 40000 116100 9c40 9C40
b=40000 40000 40000 116100 9c40 9C40
b=40000 40000 40000 116100 9c40 9C40
c=4464 4464 4464 10560 1170 1170
c=70000 70000 70000 210560 11170 11170
c=70000 70000 70000 210560 11170 11170
```

参考printf的格式控制符和附加格式控制符，给出解释：

附加控制符l的作用：

可以使数据以长整型的形式输出

附加控制符h的作用：

可以使数据以短整型的形式输出

★ 在C方式中，如果要输出的数据类型与格式控制符的类型不一致，则以格式控制符(数据类型/格式控制符)为准

提醒：先看清楚，是字母l还是数字1



# §. 基础知识题 - C方式输入输出的格式化控制

## 1. 格式化输出函数printf的基本理解

E. 观察下列程序的运行结果，回答问题并将程序的运行结果截图贴上(如果有错则贴错误信息截图)

```
#include <stdio.h>

int main()
{
    int a = 70000;
    printf("a=%ld*\n", a);
    printf("a=%10ld*\n", a);
    printf("a=%-10ld*\n\n", a);

    printf("a=%d*\n", a);
    printf("a=%10d*\n", a);
    printf("a=%10d*\n", -a);
    printf("a=%-10d*\n\n", a);
    printf("a=%-10d*\n", -a);

    printf("a=%hd*\n", a);
    printf("a=%10hd*\n", a);
    printf("a=%-10hd*\n\n", a);

    return 0;
} //注：最后加*的目的，是为了看清是否有隐含空格
```

```
Microsoft Visual Studio
a=70000*
a=      70000*
a=70000  *
a=70000*
a=      70000*
a=     -70000*
a=70000  *
a=-70000  *
a=4464*
a=      4464*
a=4464  *
a=70000*
a=      70000*
a=     -70000*
a=70000  *
a=-70000  *
a=4464*
a=      4464*
a=4464  *
```

参考printf的格式控制符和附加格式控制符，给出解释：

%ld : 以 长整型 类型的数据类型输出

%10ld : 以 长整型 类型输出，总宽度 10，右 对齐

%-10ld: 以 长整型 类型输出，总宽度 10，左 对齐

%d : 以 带符号的十进制整数 (int) 类型的数据类型输出

%10d : 以 带符号的十进制整数 (int) 类型输出，总宽度 11，右 对齐

%-10d: 以 带符号的十进制整数 (int) 类型输出，总宽度 11，左 对齐

%hd : 以 短整型 类型的数据类型输出

%10hd : 以 短整型 类型输出，总宽度 10，右 对齐

%-10hd: 以 短整型 类型输出，总宽度 10，左 对齐

如果输出负数且指定宽度，负号 占 (占/不占) 总宽度



# §. 基础知识题 - C方式输入输出的格式化控制

## 1. 格式化输出函数printf的基本理解

F. 观察下列程序的运行结果，回答问题并将程序的运行结果截图贴上(如果有错则贴错误信息截图)

```
#include <stdio.h>

int main()
{
    float f = 123.456f;
    printf("f=%f\n", f);
    printf("f=%e\n", f);
    printf("f=%E\n", f);
    printf("f=%g\n", f);
    printf("f=%G\n\n", f);

    f = 0.123456789f;
    printf("f=%f\n", f);
    printf("f=%e\n", f);
    printf("f=%E\n", f);
    printf("f=%g\n", f);
    printf("f=%G\n\n", f);

    f = 123456789.0f;
    printf("f=%f\n", f);
    printf("f=%e\n", f);
    printf("f=%E\n", f);
    printf("f=%g\n", f);
    printf("f=%G\n\n", f);

    return 0;
}
```

```
Microsoft Visual Studio 调试
f=123.456001
f=1.234560e+02
f=1.234560E+02
f=123.456
f=123.456
f=0.123457
f=1.234568e-01
f=1.234568E-01
f=0.123457
f=0.123457
f=123456792.000000
f=1.234568e+08
f=1.234568E+08
f=1.23457e+08
f=1.23457E+08
```

运行结果:

参考printf的格式控制符和附加格式控制符，给出解释:

%f: 将浮点数以十进制的 小数 形式输出

%e: 将浮点数以十进制的 指数 (e小写) 形式输出

%E: 将浮点数以十进制的 指数 (E大写) 形式输出,

%e和%E的区别是 指数e的大小写不同

%g/%G: 输出形式为 从f、e中选取宽度较短的形式

★ 仔细观察并叙述清楚，如果觉得左例还不足以理解，可以自己再构造测试数据

%g/%G: 输出形式的差别为 如果选择指数形式，则g中e小写，G中E大写。





# §. 基础知识题 - C方式输入输出的格式化控制

## 1. 格式化输出函数printf的基本理解

G. 观察下列程序的运行结果，回答问题并将程序的运行结果截图贴上(如果有错则贴错误信息截图)

```
#include <stdio.h>
int main()
{
    double f = 123.456;
    printf("f=%f\n", f);
    printf("f=%lf\n", f);
    printf("f=%e\n", f);
    printf("f=%le\n", f);
    printf("f=%g\n", f);
    printf("f=%lg\n\n", f);

    f = 0.123456789;
    printf("f=%f\n", f);
    printf("f=%lf\n", f);
    printf("f=%e\n", f);
    printf("f=%le\n", f);
    printf("f=%g\n", f);
    printf("f=%lg\n\n", f);

    f = 123456789.0;
    printf("f=%f\n", f);
    printf("f=%lf\n", f);
    printf("f=%e\n", f);
    printf("f=%le\n", f);
    printf("f=%g\n", f);
    printf("f=%lg\n\n", f);
    return 0;
}
```

```
Microsoft Visual Studio 调试
f=123.456000
f=123.456000
f=1.234560e+02
f=1.234560e+02
f=123.456
f=123.456

f=0.123457
f=0.123457
f=1.234568e-01
f=1.234568e-01
f=0.123457
f=0.123457

f=123456789.000000
f=123456789.000000
f=1.234568e+08
f=1.234568e+08
f=1.23457e+08
f=1.23457e+08
```

运行结果:

参考printf的格式控制符和附加格式控制符，给出解释:

对于double数据:

1、格式符%f和%lf是否有区别?

没有区别. (因为float型和long型均为4字节)

2、如何证明你给出的1的结论?

(提示: 三组数据的哪组能证明?)

第二组数据可以证明。

由于double为8字节，向四字节的float和long转换时会截断，由第二组的输出结果可知道最后截断的结果是相同的。



# § . 基础知识题 - C方式输入输出的格式化控制

## 1. 格式化输出函数printf的基本理解

H. 观察下列程序的运行结果，回答问题并将程序的运行结果截图贴上(如果有错则贴错误信息截图)

```
#include <stdio.h>

int main()
{
    double f = 123456.789;

    printf("f=%f*\n", f);
    printf("f=%.2f*\n", f);
    printf("f=%10.2f*\n", f);
    printf("f=%-10.2f*\n\n", f);

    printf("f=%e*\n", f);
    printf("f=%.2e*\n", f);
    printf("f=%10.2e*\n", f);
    printf("f=%-10.2e*\n\n", f);

    printf("f=%g*\n", f);
    printf("f=%.2g*\n", f);
    printf("f=%.3g*\n", f);
    printf("f=%10.2g*\n", -f);
    printf("f=%10.3g*\n", f);
    printf("f=%-10.2g*\n", -f);
    printf("f=%-10.3g*\n", f);

    return 0;
}
```

//注：最后加\*的目的，是为了看清是否有隐含空格

```
Microsoft Visual Studio 调试
f=123456.789000*
f=123456.79*
f= 123456.79*
f=123456.79 *
f=1.234568e+05*
f=1.23e+05*
f= 1.23e+05*
f=1.23e+05 *
f=123457*
f=1.2e+05*
f=1.23e+05*
f= -1.2e+05*
f= 1.23e+05*
f=-1.2e+05 *
f=1.23e+05 *
```

运行结果：

参考printf的格式控制符和附加格式控制符，给出解释：

%10.2f : 以 float 类型输出，总宽度 10，  
小数点后 2 位，右 对齐

%-10.2f: 以 float 类型输出，总宽度 10，  
小数点后 2 位，左 对齐

%10.2e : 以 float指数 类型输出，总宽度 10，  
小数点后 2 位，右 对齐

%-10.2e: 以 float指数 类型输出，总宽度 10，  
小数点后 2 位，左 对齐

对%f和%e而言，指定的总宽度包含 (包含/不包含)小数点

对%g而言，%m.n中n代表的位数是指 有效数字位数

如果输出负数且指定宽度，负号占 (占/不占)总宽度



# § . 基础知识题 - C方式输入输出的格式化控制

## 1. 格式化输出函数printf的基本理解

I. 观察下列程序的运行结果，回答问题并将程序的运行结果截图贴上(如果有错则贴错误信息截图)

```
#include <stdio.h>

int main()
{
    float f = 123456789.123;

    printf("f=%f*\n", f);
    printf("f=%10.2f*\n", f);
    printf("f=%-10.2f*\n", f);
    printf("f=%.2f*\n\n", f);

    double d = 12345678901234567.6789;

    printf("d=%f*\n", d);
    printf("d=%10.2f*\n", d);
    printf("d=%-10.2f*\n", d);
    printf("d=%.2f*\n\n", d);

    return 0;
}
```

**//注：最后加\*的目的，是为了看清是否有隐含空格**

运行结果：

```
Microsoft Visual Studio 调试控制台
f=123456792.000000*
f=123456792.00*
f=123456792.00*
f=123456792.00*
d=12345678901234568.000000*
d=12345678901234568.00*
d=12345678901234568.00*
d=12345678901234568.00*
```

给出下面两个概念的结论：

- 1、在数据的有效位数超过精度时，则输出：  
在原数据基础上超出有效位数的值置为不可信值，并用0填充六位小数。
- 2、如果指定的总宽度小于有效位数的宽度，则输出：  
忽略总宽度的限制，输出数据（但该输出数据整数部分是在原数据基础上超出有效位数的值置为不可信值，小数部分用0填充）



# § . 基础知识题 - C方式输入输出的格式化控制

## 1. 格式化输出函数printf的基本理解

J. 观察下列程序的运行结果，回答问题并将程序的运行结果截图贴上(如果有错则贴错误信息截图)

```
#include <stdio.h>

#define str "abcdefghijklmnopqrstuvwxy*"

int main()
{
    printf("str=%s*\n", str);
    printf("str=%30s*\n", str);
    printf("str=%-30s*\n", str);
    printf("str=%5s*\n", str);
    printf("str=%-5s*\n", str);
    printf("str=%.5s*\n", str);
    printf("str=%-.5s*\n", str);
    printf("str=%10.5s*\n", str);
    printf("str=%-10.5s*\n", str);

    return 0;
}
```

**//注：最后加\*的目的，是为了看清是否有隐含空格**

运行结果：



参考printf的格式控制符和附加格式控制符，给出解释：

- %s : 输出 字符串 类型的数据
- %30s : 输出 字符串 类型的数据，总宽度 30，右 对齐
- %-30s: 输出 字符串 类型的数据，总宽度 30，左 对齐

如果指定的总宽度小于字符串的长度，则：  
按照指定宽度截取原字符串并输出  
对%s而言，%m.n中n代表的位数是指 输出字符串中字符的个数



## §. 基础知识题 - C方式输入输出的格式化控制

### 1. 格式化输出函数printf的基本理解

K. 观察下列程序的运行结果，回答问题并将程序的运行结果截图贴上(如果有错则贴错误信息截图)

```
#include <stdio.h>

#define str "Student"
int main()
{
    int a = 65;
    printf("a=%o\n", a);
    printf("a=%x\n", a);
    printf("ch=%c\n", a);
    printf("s=%s\n\n", str);

    printf("a=0%o\n", a);
    printf("a=0x%x\n", a);
    printf("ch=\'%c\' \n", a);
    printf("s=\'"%s"\' \n\n", str);

    double d = 0.783;
    printf("百分比=%.2f%%\n", d * 100);

    return 0;
}
```

运行结果:

```
Microsoft Visual Studio 调试
a=101
a=41
ch=A
s=Student

a=0101
a=0x41
ch='A'
s='\"Student\"'

百分比=78.30%
```

1、对比第1组和第2组输出，得出的结论是：

格式控制符/附加格式控制符，只负责给出\_\_原始数据\_\_的输出，若需要前导字符、单双引号等，需要\_\_自行添加\_\_

2、输出字符'%'的方法是: “%%”





# § . 基础知识题 - C方式输入输出的格式化控制

## 2. 格式化输入函数scanf的基本理解

形式：scanf (格式控制表列，地址表列)；

格式控制表列的内容：

格式说明：以%开始+格式字符，表示按格式输入

普通字符(含转义符)：原样输入

地址表列：

&表示取地址

&变量名：取该变量的内存地址

★ &不能跟表达式/常量(理由与=、++、--等相同)

常用的格式符种类：

scanf所用的格式字符的种类：

d, i	输入带符号的十进制形式整数
o	输入八进制无符号形式整数(不带前导0)
x, X	输入十六进制无符号形式整数(不带前导0x)
u	输入十进制无符号形式整数
c	输入单个字符
s	输入字符串
f	输入小数/指数形式的浮点数
e, E, g, G	同f

特别说明：

VS系列认为scanf函数是不安全的输入，因此缺省禁止使用(编译报error)，如果想继续使用，必须在源程序一开始加定义

```
#define _CRT_SECURE_NO_WARNINGS
```

为了和其它编译器兼容，以及方便后续课程的学习，我们仍然会继续使用scanf

另：加 \_CRT\_SECURE\_NO\_WARNINGS 的程序在其它编译器中可正常使用

注：VS系列中C语言用于安全输入的函数是scanf\_s，使用方法同scanf，考虑到兼容性，不建议大家使用scanf\_s，有兴趣可以自行查阅有关资料

scanf所用的附加格式字符的种类：

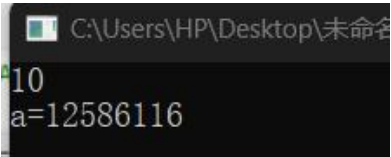
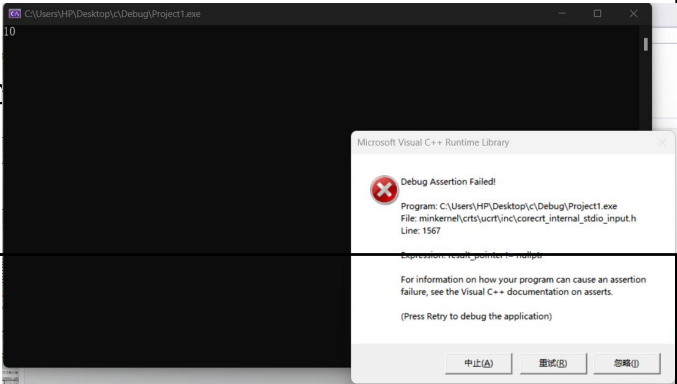
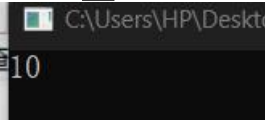
字母l	输入长整型数，用于d, o, x, u前 输入double型数，用于f, e, g前
h	输入短整型数，用于d, o, x, u前
正整数n	指定输入数据所占的宽度
*	本输入项不赋给相应的变量



# § . 基础知识题 - C方式输入输出的格式化控制

## 2. 格式化输入函数scanf的基本理解

A. 观察下列程序的运行结果，回答问题并将程序的运行结果截图贴上(如果有错则贴错误信息截图)

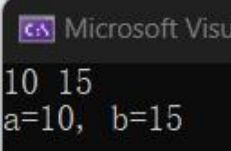
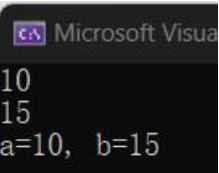
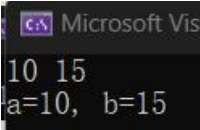
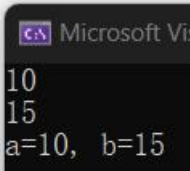
<pre>#define _CRT_SECURE_NO_WARNINGS #include &lt;stdio.h&gt;  int main() {     int a;     scanf("%d", a);     printf("a=%d\n", a);     return 0; }</pre>	<pre>#define _CRT_SECURE_NO_WARNINGS #include &lt;stdio.h&gt;  int main() {     int a = 0;     scanf("%d", a)     printf("a=%d\n", a);     return 0; }</pre>
<p>在VS中编译:</p> <p>warning C4477: “scanf”: 格式字符串 “%d” 需要类型 “int *” 的参数, 但可变参数 1 拥有了类型 “int” error C4700: 使用了未初始化的局部变量 “a”</p> <p>在Dev中编译: 假设键盘输入为: 10✓ (✓表示回车键, 下同) 则输出为:</p> 	<p>在VS中编译: 假设键盘输入为: 10✓ 则输出为:</p>  <p>在Dev中编译: 假设键盘输入为: 10✓ 则输出为:</p>  <p>结论: 用scanf输入时, 如果地址表列中直接跟变量名, 则 <u>错误</u> (错误/正确), 其中VS的表现是 <u>无法运行</u>, Dev的表现是 <u>能运行, 但结果错误</u></p>



# § . 基础知识题 - C方式输入输出的格式化控制

## 2. 格式化输入函数scanf的基本理解

B. 观察下列程序的运行结果，回答问题并将程序的运行结果截图贴上(如果有错则贴错误信息截图)

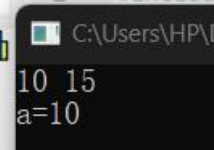

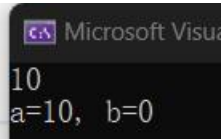
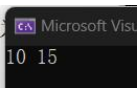
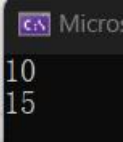
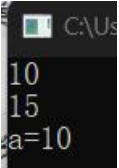
<pre>#define _CRT_SECURE_NO_WARNINGS #include &lt;stdio.h&gt;  int main() {     int a, b;     scanf("%d %d", &amp;a, &amp;b);     printf("a=%d, b=%d\n", a, b);     return 0; }</pre>	<pre>#define _CRT_SECURE_NO_WARNINGS #include &lt;stdio.h&gt;  int main() {     int a, b;     scanf("%d%d", &amp;a, &amp;b); //%d间无空格     printf("a=%d, b=%d\n", a, b);     return 0; }</pre>
<p>假设键盘输入为: 10 15✓ 则输出为: </p> <p>假设键盘输入为: 10✓ 15✓ 则输出为: </p>	<p>假设键盘输入为: 10 15✓ 则输出为: </p> <p>假设键盘输入为: 10✓ 15✓ 则输出为: </p> <p>结论: 多个输入时, 格式控制符间是否有空格<u>不影响</u> (影响/不影响)正确性</p>



# § . 基础知识题 - C方式输入输出的格式化控制

## 2. 格式化输入函数scanf的基本理解

C. 观察下列程序的运行结果，回答问题并将程序的运行结果截图贴上(如果有错则贴错误信息截图)

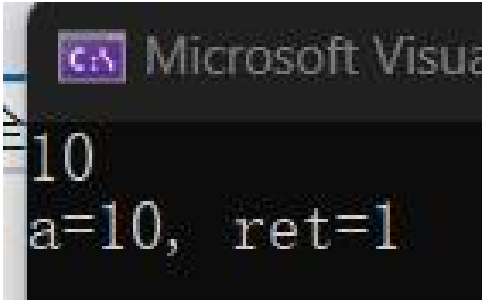
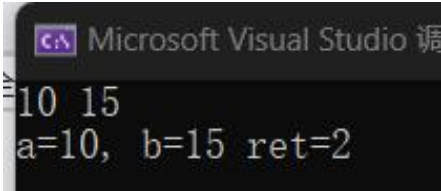
<pre>#define _CRT_SECURE_NO_WARNINGS #include &lt;stdio.h&gt;  int main() {     int a=0, b=0;     scanf("%d", &amp;a, &amp;b); //地址表列多     printf("a=%d, b=%d\n", a, b);     return 0; }</pre>		<pre>#define _CRT_SECURE_NO_WARNINGS #include &lt;stdio.h&gt;  int main() {     int a;     scanf("%d %d", &amp;a); //格式符多     printf("a=%d\n", a);     return 0; }</pre> <div>VS报warning: Users\HP\Desktop\c\Project1\111.c(7,11): warning C4473: "scanf": 没有为格式字符串传递足够的参数 C:\Users\HP\Desktop\c\Project1\111.c(7,11): 占位符和其参数预计 2 可变参数, 但提供的却是 1 参数 C:\Users\HP\Desktop\c\Project1\111.c(7,11): 缺失的可变参数 2 为格式字符串 "%d" 所需</div> 	
<p>假设键盘输入为: <u>10 15</u>✓ 则输出为:</p>  <p>假设键盘输入为: <u>10</u>✓ 则输出为:</p>  <p>结论: 当地址表列的个数多于格式控制符时, <u>只会读取一个输入的数字赋给一个变量</u></p>		<p>VS: 假设键盘输入为: <u>10 15</u>✓ 则输出为:</p>  <p>假设键盘输入为: <u>10</u>✓ <u>15</u>✓ 则输出为:</p>  <p>Dev: 假设键盘输入为: <u>10 15</u>✓ 则输出为:</p>  <p>结论: 当格式控制符的个数多个地址表列时 <u>VS报warning, 执行无结果; Dev只会读取一个输入的数字, 赋给一个变量</u></p>	



# § . 基础知识题 - C方式输入输出的格式化控制

## 2. 格式化输入函数scanf的基本理解

D. 观察下列程序的运行结果，回答问题并将程序的运行结果截图贴上(如果有错则贴错误信息截图)

<pre>#define _CRT_SECURE_NO_WARNINGS #include &lt;stdio.h&gt;  int main() {     int a, ret;     ret = scanf("%d", &amp;a);     printf("a=%d, ret=%d\n", a, ret);     return 0; }</pre>	<pre>#define _CRT_SECURE_NO_WARNINGS #include &lt;stdio.h&gt;  int main() {     int a, b, ret;     ret = scanf("%d %d", &amp;a, &amp;b);     printf("a=%d, b=%d ret=%d\n", a, b, ret);     return 0; }</pre>
<p>假设键盘输入为: <u>10</u>✓ 则输出为:</p> 	<p>假设键盘输入为: <u>10 15</u>✓ 则输出为:</p>  <p>结论: 在输入正确时, scanf的返回值是__<u>正确输入成功读取的参数个数</u>__</p>

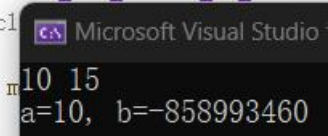
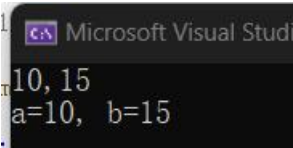
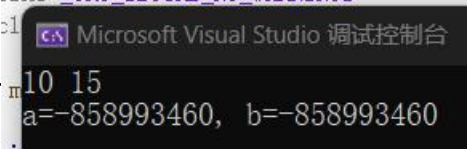
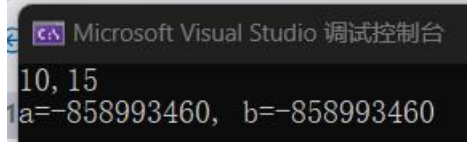
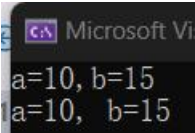




# § . 基础知识题 - C方式输入输出的格式化控制

## 2. 格式化输入函数scanf的基本理解

E. 观察下列程序的运行结果，回答问题并将程序的运行结果截图贴上(如果有错则贴错误信息截图)

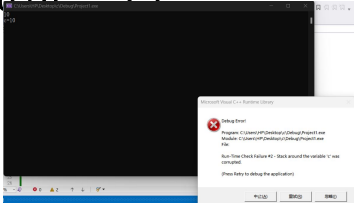
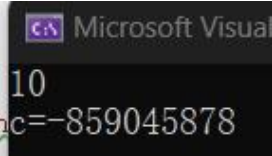
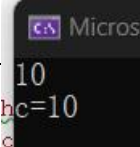
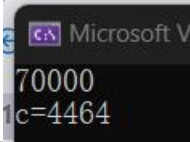
<pre>#define _CRT_SECURE_NO_WARNINGS #include &lt;stdio.h&gt;  int main() {     int a, b;     scanf("%d,%d", &amp;a, &amp;b);     printf("a=%d, b=%d\n", a, b);     return 0; }</pre>	<pre>#define _CRT_SECURE_NO_WARNINGS #include &lt;stdio.h&gt;  int main() {     int a, b;     scanf("a=%d,b=%d", &amp;a, &amp;b);     printf("a=%d, b=%d\n", a, b);     return 0; }</pre>
<p>假设键盘输入为: <u>10 15</u>✓ 则输出为:</p> <p>假设键盘输入为: <u>10,15</u>✓ 则输出为:</p>  	<p>假设键盘输入为: <u>10 15</u>✓ 则输出为:</p>  <p>假设键盘输入为: <u>10,15</u>✓ 则输出为:</p>  <p>假设键盘输入为: <u>a=10,b=15</u>✓ 则输出为:</p>  <p>结论: 当格式控制符中有其它字符(逗号, a=等)时, 对这些字符的输入方法是 <u>直接作为输入的内容</u></p>



# § . 基础知识题 - C方式输入输出的格式化控制

## 2. 格式化输入函数scanf的基本理解

F. 观察下列程序的运行结果，回答问题并将程序的运行结果截图贴上(如果有错则贴错误信息截图)

<pre>#define _CRT_SECURE_NO_WARNINGS #include &lt;stdio.h&gt;  int main() {     short c;      scanf("%d", &amp;c);     printf("c=%hd\n", c);      return 0; }</pre> 	<pre>#define _CRT_SECURE_NO_WARNINGS #include &lt;stdio.h&gt;  int main() {     int c;      scanf("%hd", &amp;c);     printf("c=%d\n", c);      return 0; }</pre>	<pre>#define _CRT_SECURE_NO_WARNINGS #include &lt;stdio.h&gt;  int main() {     short c;      scanf("%hd", &amp;c);     printf("c=%hd\n", c);      return 0; }</pre>
<p>假设键盘输入为: <u>10</u>✓ 则输出为: <u>c=10, 但是会弹窗报错</u></p>	<p>假设键盘输入为: <u>10</u>✓ 则输出为:</p> 	<p>假设键盘输入为: <u>10</u>✓ 则输出为:</p>  <p>假设键盘输入为: <u>70000</u>✓ 则输出为:</p> 
<p>结论:</p> <p>1、附加格式控制符h的作用是 <u>输入短整型数</u></p> <p>2、如果格式控制符的数据类型和要读取的变量类型的字节大小不一致 (例: 4/2字节), 则 <u>存储的数据是截断后的数据</u></p> <p>3、记住这个page, 相关错误的原理性分析, 第6章完成后会明白!!!</p>		



## §. 基础知识题 - C方式输入输出的格式化控制

### 2. 格式化输入函数scanf的基本理解

G. 观察下列程序的运行结果，回答问题并将程序的运行结果截图贴上(如果有错则贴错误信息截图)

```
#define _CRT_SECURE_NO_WARNINGS
#include <stdio.h>

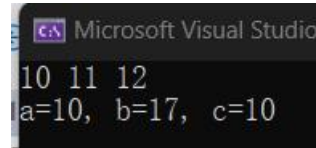
int main()
{
    int a, b, c;

    scanf("%d %x %o", &a, &b, &c);
    printf("a=%d, b=%d, c=%d\n", a, b, c);

    return 0;
}
```

假设键盘输入为: 10 11 12✓

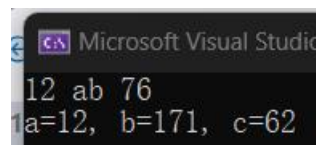
则输出为:



```
Microsoft Visual Studio
10 11 12
a=10, b=17, c=10
```

假设键盘输入为: 12 ab 76✓

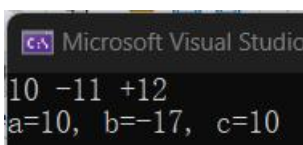
则输出为:



```
Microsoft Visual Studio
12 ab 76
a=12, b=171, c=62
```

假设键盘输入为: 10 -11 +12✓

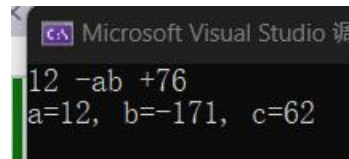
则输出为:



```
Microsoft Visual Studio
10 -11 +12
a=10, b=-17, c=10
```

假设键盘输入为: 12 -ab +76✓

则输出为:



```
Microsoft Visual Studio 调
12 -ab +76
a=12, b=-171, c=62
```



# § . 基础知识题 - C方式输入输出的格式化控制

## 2. 格式化输入函数scanf的基本理解

H. 观察下列程序的运行结果，回答问题并将程序的运行结果截图贴上(如果有错则贴错误信息截图)

```
#define _CRT_SECURE_NO_WARNINGS
#include <stdio.h>

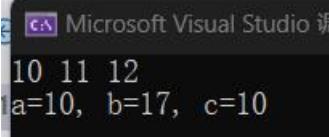
int main()
{
    short a, b, c;

    scanf("%hd %hx %ho", &a, &b, &c);
    printf("a=%hd, b=%hd, c=%hd\n", a, b, c);

    return 0;
}
```

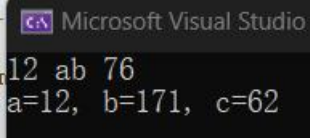
假设键盘输入为: 10 11 12✓

则输出为:



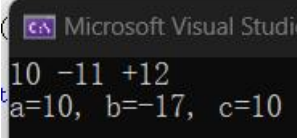
假设键盘输入为: 12 ab 76✓

则输出为:



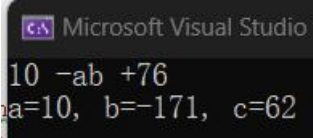
假设键盘输入为: 10 -11 +12✓

则输出为:



假设键盘输入为: 12 -ab +76✓

则输出为:

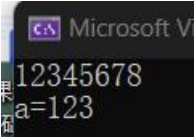
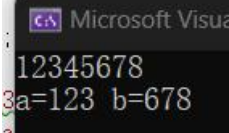




# § . 基础知识题 - C方式输入输出的格式化控制

## 2. 格式化输入函数scanf的基本理解

I. 观察下列程序的运行结果，回答问题并将程序的运行结果截图贴上(如果有错则贴错误信息截图)

<pre>#define _CRT_SECURE_NO_WARNINGS #include &lt;stdio.h&gt;  int main() {     int a;      scanf("%3d", &amp;a);     printf("a=%d\n", a);      return 0; }</pre>	<pre>#define _CRT_SECURE_NO_WARNINGS #include &lt;stdio.h&gt;  int main() {     int a, b;      scanf("%3d %*2d %3d", &amp;a, &amp;b);     printf("a=%d b=%d\n", a, b);      return 0; }</pre>
<p>假设键盘输入为: <u>12345678</u>✓ 则输出为:</p>  <p>结论: %md中的m表示: <u>读取的输入字符个数</u></p>	<p>假设键盘输入为: <u>12345678</u>✓ 则输出为:</p>  <p>结论: *md的*m表示: <u>跳过的输入字符个数 (即读取前三个, 跳过第四个和第五个不读取赋值, 再读取三个)</u></p>

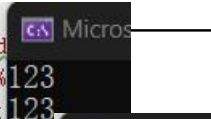
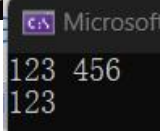

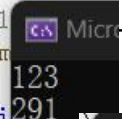

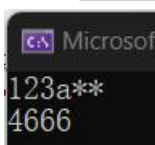

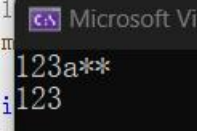





# § . 基础知识题 - C方式输入输出的格式化控制

## 2. 格式化输入函数scanf的基本理解

J. 观察下列程序的运行结果，回答问题并将程序的运行结果截图贴上(如果有错则贴错误信息截图)

<pre>#define _CRT_SECURE_NO_WARNINGS #include &lt;stdio.h&gt; int main() {     int a;     scanf("%d", &amp;a);     printf("%d\n", a);     return 0; }</pre>	<pre>#define _CRT_SECURE_NO_WARNINGS #include &lt;stdio.h&gt; int main() {     int a;     scanf("%x", &amp;a);     printf("%d\n", a);     return 0; }</pre>	<pre>#define _CRT_SECURE_NO_WARNINGS #include &lt;stdio.h&gt; int main() {     int a;     scanf("%3d", &amp;a);     printf("%d\n", a);     return 0; }</pre>
<p>假设键盘输入为: <u>123</u>✓ 则输出为:</p>  <p>假设键盘输入为: <u>123 456</u>✓ 则输出为:</p>  <p>假设键盘输入为: <u>123a**</u>✓ 则输出为:</p> 	<p>假设键盘输入为: <u>123</u>✓ 则输出为:</p>  <p>假设键盘输入为: <u>123 456</u>✓ 则输出为:</p>  <p>假设键盘输入为: <u>123a**</u>✓ 则输出为:</p> 	<p>假设键盘输入为: <u>123</u>✓ 则输出为:</p>  <p>假设键盘输入为: <u>123a**</u>✓ 则输出为:</p>  <p>假设键盘输入为: <u>12a**</u>✓ 则输出为:</p> 
<p>结论: scanf输入的终止条件是<u>空格</u>、<u>与格式说明符不符的字符（非法字符）</u>、<u>超出指定宽度</u>和<u>回车</u>（共四项）</p>		



# § . 基础知识题 - C方式输入输出的格式化控制

## 2. 格式化输入函数scanf的基本理解

K. 观察下列程序的运行结果，回答问题并将程序的运行结果截图贴上(如果有错则贴错误信息截图)

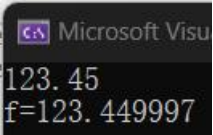
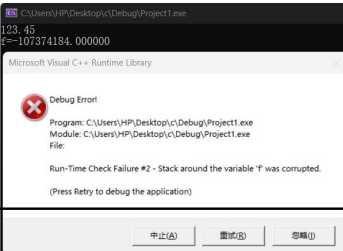
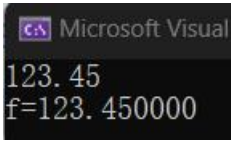

<pre>#define _CRT_SECURE_NO_WARNINGS #include &lt;stdio.h&gt; int main() {     int a, b;     scanf("%3d%3d", &amp;a, &amp;b);     printf("%d %d\n", a, b);     return 0; }</pre>	<pre>#define _CRT_SECURE_NO_WARNINGS #include &lt;stdio.h&gt; int main() {     int a, b;     scanf("%3d*2d%3d", &amp;a, &amp;b);     printf("%d %d\n", a, b);     return 0; }</pre>
<p>输入: 12 345 , 输出: 12 345</p> <p>输入: 12 3456 , 输出: 12 3456</p> <p>输入: 123 456 , 输出: 123 456</p> <p>输入: 1234 5678 , 输出: 1234 5678</p> <p>输入: 123456 , 输出: 123456</p> <p>输入: 12345678 , 输出: 12345678</p> <p>注: 特别关注第4项的结果, 想想为什么?</p>	<p>输入: 123456 , 输出: 123456</p> <p>输入: 12345678 , 输出: 12345678</p> <p>输入: 123456789 , 输出: 123456789</p> <p>输入: 123 45 678 , 输出: 123 45 678</p>
<p>说明键入的字符存储在缓冲区, 需要的时候再取</p> <p>考查上题得出的scanf终止条件的结论是否完整, 如果不完整, 补充修改上题的结论</p>	



# § . 基础知识题 - C方式输入输出的格式化控制

## 2. 格式化输入函数scanf的基本理解

L. 观察下列程序的运行结果，回答问题并将程序的运行结果截图贴上(如果有错则贴错误信息截图)

<pre>#define _CRT_SECURE_NO_WARNINGS #include &lt;stdio.h&gt;  int main() {     float f;      scanf("%f", &amp;f);     printf("f=%f\n", f);      return 0; }</pre>	<pre>#define _CRT_SECURE_NO_WARNINGS #include &lt;stdio.h&gt;  int main() {     float f;      scanf("%lf", &amp;f);     printf("f=%f\n", f);      return 0; }</pre>	<pre>#define _CRT_SECURE_NO_WARNINGS #include &lt;stdio.h&gt;  int main() {     double f;      scanf("%lf", &amp;f);     printf("f=%f\n", f);      return 0; }</pre>	<pre>#define _CRT_SECURE_NO_WARNINGS #include &lt;stdio.h&gt;  int main() {     double f;      scanf("%f", &amp;f);     printf("f=%f\n", f);      return 0; }</pre>
假设键盘输入为: 123.45✓ 则输出为: 	假设键盘输入为: 123.45✓ 则输出为: 	假设键盘输入为: 123.45✓ 则输出为: 	假设键盘输入为: 123.45✓ 则输出为: 

结论:

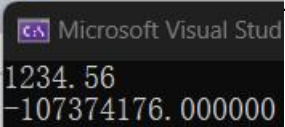
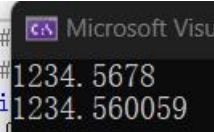
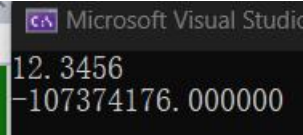
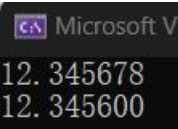
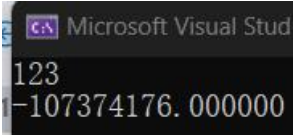
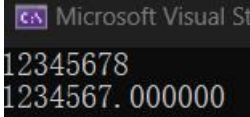
- 1、附加格式控制符l的作用是 输入长整型数或double型数据
- 2、如果格式控制符的数据类型和要读取的变量类型的字节大小不一致（例：4/8字节）则 会报错或直接数据截断
- 3、printf中，输出double型数据时，%f 和 %lf 无（有/无）差别；  
scanf中，输入double型数据时，%f 和 %lf 有（有/无）差别



# § . 基础知识题 - C方式输入输出的格式化控制

## 2. 格式化输入函数scanf的基本理解

M. 观察下列程序的运行结果，回答问题并将程序的运行结果截图贴上(如果有错则贴错误信息截图)

<pre>#define _CRT_SECURE_NO_WARNINGS #include &lt;stdio.h&gt; int main() {     float f;     scanf("%7.2f", &amp;f);     printf("%f\n", f);     return 0; }</pre>	<pre>#define _CRT_SECURE_NO_WARNINGS #include &lt;stdio.h&gt; int main() {     float f;     scanf("%7f", &amp;f);     printf("%f\n", f);     return 0; }</pre>
<p>假设键盘输入为: <u>1234.56</u>✓ 则输出为:</p> 	<p>假设键盘输入为: <u>1234.5678</u>✓ 则输出为:</p> 
<p>假设键盘输入为: <u>12.3456</u>✓ 则输出为:</p> 	<p>假设键盘输入为: <u>12.345678</u>✓ 则输出为:</p> 
<p>假设键盘输入为: <u>123</u>✓ 则输出为:</p> 	<p>假设键盘输入为: <u>12345678</u>✓ 则输出为:</p> 
<p>结论:</p> <p>1、%mf/%mLf如果指定了宽度m, 则 <u>输出结果中超过宽度m的值为不可信值</u></p> <p>2、%m.nf/%m.nLf如果指定了精度(小数点后的位数), 则 <u>无效(因为不支持该格式控制)</u> (注: 确认scanf的%f/%Lf是否支持.n形式的附加格式控制符!!!)</p>	



# § . 基础知识题 - C方式输入输出的格式化控制

## 2. 格式化输入函数scanf的基本理解

N. 观察下列程序的运行结果，回答问题并将程序的运行结果截图贴上(如果有错则贴错误信息截图)

<pre>#define _CRT_SECURE_NO_WARNINGS #include &lt;stdio.h&gt; int main() {     char c1, c2;     scanf("%c %c", &amp;c1, &amp;c2);     printf("c1=%c c2=%c\n", c1, c2);     return 0; }</pre>	<pre>#define _CRT_SECURE_NO_WARNINGS #include &lt;stdio.h&gt; int main() {     char c1, c2;     scanf("%c%c", &amp;c1, &amp;c2); //两个%c间无空格     printf("c1=%d c2=%d\n", c1, c2);     return 0; }</pre>
<p>假设键盘输入为: <u>ABCD</u>✓ 则输出为:</p> <p>假设键盘输入为: <u>A BCD</u>✓ 则输出为:</p> <p>假设键盘输入为: <u>'A' BCD</u>✓ 则输出为:</p> <p>假设键盘输入为: <u>\n</u>✓ 则输出为:</p>	<p>假设键盘输入为: <u>ABCD</u>✓ 则输出为:</p> <p>假设键盘输入为: <u>A BCD</u>✓ (特别关注此项的差异) 则输出为:</p> <p>假设键盘输入为: <u>'A' BCD</u>✓ 则输出为:</p> <p>假设键盘输入为: <u>\n</u>✓ 则输出为:</p>
<p>结论:</p> <p>1、%c只读<u>1</u>个字符</p> <p>2、%c在输入转义符/单引号等特殊字符时，得到的是<u>特殊字符自身的 ASCII码</u> (特殊字符自身的ASCII码/特殊字符的转义含义)</p> <p>3、空格<u>是</u> (是/不是)scanf中%c方式的有效输入，但必须注意<u>在%c之前添加空格可以使scanf跳过空格</u></p>	

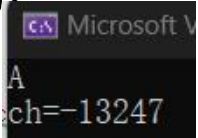
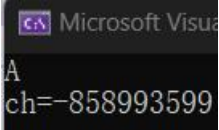
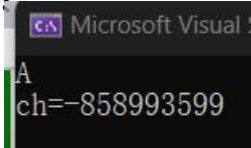
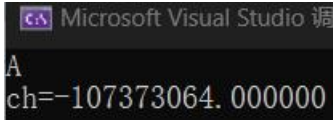




# § . 基础知识题 - C方式输入输出的格式化控制

## 2. 格式化输入函数scanf的基本理解

0. 观察下列程序的运行结果，回答问题并将程序的运行结果截图贴上(如果有错则贴错误信息截图)

<pre>#define _CRT_SECURE_NO_WARNINGS #include &lt;stdio.h&gt;  int main() {     short ch;      scanf("%c", &amp;ch);     printf("ch=%hd\n", ch);      return 0; }</pre>	<pre>#define _CRT_SECURE_NO_WARNINGS #include &lt;stdio.h&gt;  int main() {     int ch;      scanf("%c", &amp;ch);     printf("ch=%d\n", ch);      return 0; }</pre>	<pre>#define _CRT_SECURE_NO_WARNINGS #include &lt;stdio.h&gt;  int main() {     long ch;      scanf("%c", &amp;ch);     printf("ch=%ld\n", ch);      return 0; }</pre>	<pre>#define _CRT_SECURE_NO_WARNINGS #include &lt;stdio.h&gt;  int main() {     float ch;      scanf("%c", &amp;ch);     printf("ch=%f\n", ch);      return 0; }</pre>
假设键盘输入为: <u>A</u> ✓ 则输出为: 	假设键盘输入为: <u>A</u> ✓ 则输出为: 	假设键盘输入为: <u>A</u> ✓ 则输出为: 	假设键盘输入为: <u>A</u> ✓ 则输出为: 

结论:  
%c方式读入时，地址表列中的变量不能是 不能直接存储单个字符的 类型(不要列short/int/long/float等具体名称，总结共性)

目前只需要记住现象/结论，学习完第6章后，会从原理上理解为什么有错!!!



# § . 基础知识题 - C方式输入输出的格式化控制

## 2. 格式化输入函数scanf的基本理解

P. 观察下列程序的运行结果，回答问题并将程序的运行结果截图贴上(如果有错则贴错误信息截图)

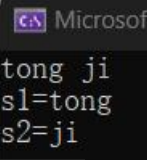
```
#define _CRT_SECURE_NO_WARNINGS
#include <stdio.h>

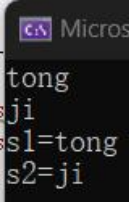
int main()
{
    char s1[10], s2[10]; //s1/s2是数组(后续内容)

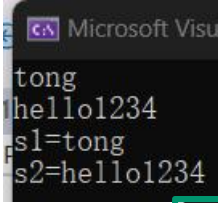
    scanf("%s %s", s1, s2);
    printf("s1=%s\ns2=%s\n", s1, s2);

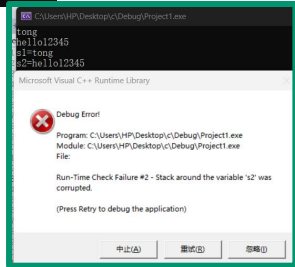
    return 0;
}
```

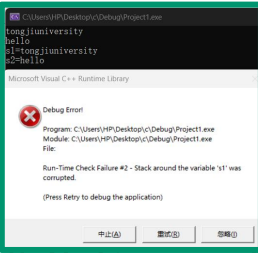
**/\* 特别说明:**  
数组名, 代表了数组的首地址, 因此放在scanf中时,  
s1/s2可以不加&, 具体概念后续数组时再详细说明  
\*/

假设键盘输入为: tong ji✓  
则输出为: 

假设键盘输入为: tong✓  
ji✓  
则输出为: 

假设键盘输入为: tong✓  
hello1234✓ (9个字符)  
则输出为: 

假设键盘输入为: tong✓  
hello12345✓ (10个字符) → 

假设键盘输入为: tongjiuniversity✓ (超过10个)  
hello✓ ← 

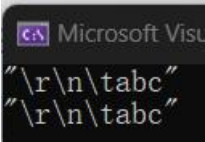
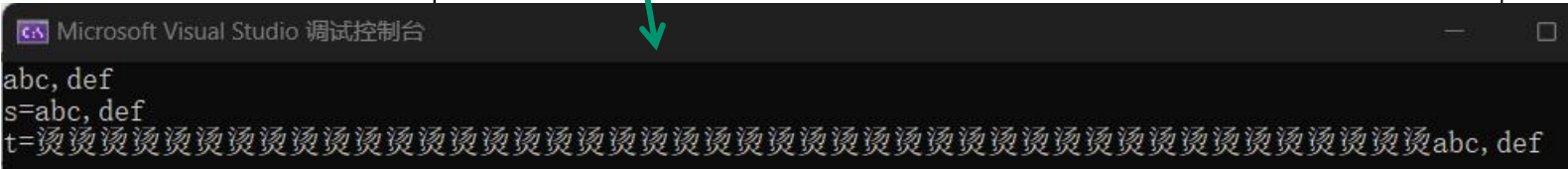
结论:  
1、%s 能 (能/不能)读入含空格的字符串  
2、%s输入时, 如果数组的大小为n, 则最多输入9个字符



# § . 基础知识题 - C方式输入输出的格式化控制

## 2. 格式化输入函数scanf的基本理解

Q. 观察下列程序的运行结果，回答问题并将程序的运行结果截图贴上(如果有错则贴错误信息截图)

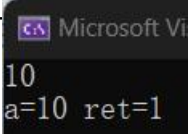
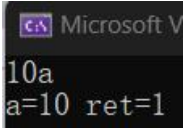
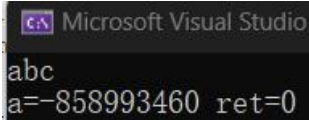
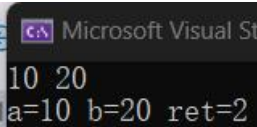
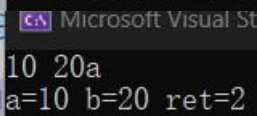
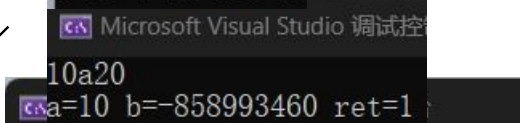
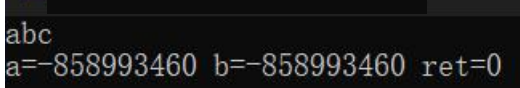
<pre>#define _CRT_SECURE_NO_WARNINGS #include &lt;stdio.h&gt; int main() {     char s[80];     scanf("%s", s);     printf("%s\n", s);     return 0; }</pre>	<pre>#define _CRT_SECURE_NO_WARNINGS #include &lt;stdio.h&gt; int main() {     char s[80], t[80];     scanf("%s,%s", s,t);     printf("s=%s\n", s);     printf("t=%s\n", t);     return 0; }</pre>
<p>假设键盘输入为: <u>"\r\n\tabc"</u> ✓ 则输出为:</p>  <p>该字符串真正的内存存储为<u>4</u>个字节，这些字节的值分别是<u>'a' (97) 'b' (98) 'c' (99)</u> <u>'\0' (0)</u></p>	<p>假设键盘输入为: <u>abc,def</u> ✓ 则输出为:</p> <p>与2-E不同，"%s,%s"之间的逗号是<u>原样输入</u> (原样输入/当做第一个字符串的有效字符)</p> 



# § . 基础知识题 - C方式输入输出的格式化控制

## 2. 格式化输入函数scanf的基本理解

R. 观察下列程序的运行结果，回答问题并将程序的运行结果截图贴上(如果有错则贴错误信息截图)

<pre>#define _CRT_SECURE_NO_WARNINGS #include &lt;stdio.h&gt; int main() {     int a, ret;     ret = scanf("%d", &amp;a);     printf("a=%d ret=%d\n", a, ret);     return 0; }</pre>	<pre>#define _CRT_SECURE_NO_WARNINGS #include &lt;stdio.h&gt; int main() {     int a, b, ret;     ret = scanf("%d %d", &amp;a, &amp;b);     printf("a=%d b=%d ret=%d\n", a, b, ret);     return 0; }</pre>
<p>假设键盘输入为: <u>10</u>✓ 则输出为:</p>  <p>假设键盘输入为: <u>10a</u>✓ 则输出为:</p>  <p>假设键盘输入为: <u>abc</u>✓ 则输出为:</p> 	<p>假设键盘输入为: <u>10 20</u>✓ 则输出为:</p>  <p>假设键盘输入为: <u>10 20a</u>✓ 则输出为:</p>  <p>假设键盘输入为: <u>10a20</u>✓ 则输出为:</p>  <p>假设键盘输入为: <u>abc</u>✓ 则输出为:</p> 
<p>结论: scanf返回值是 <u>符合输入格式并且成功赋值的参数个数</u></p>	