



## § . 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、**10月6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

```
Microsoft Visual Studio 调试控制台  
Hello, world!  
D:\Workspace\VS2019-Demo\Debug\cpp-demo.exe (进程 7484)已退出, 代码为 0。  
按任意键关闭此窗口. . .
```

例：有效贴图

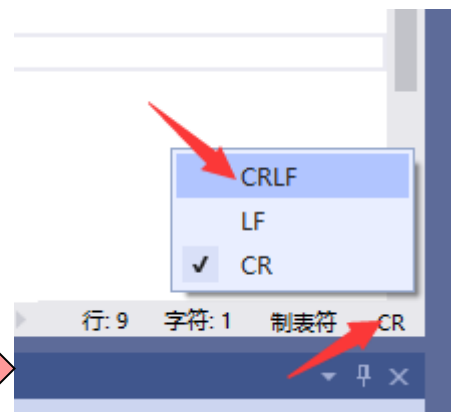
```
Microsoft Visual Studio 调试控制台  
Hello, world!
```



## §. 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?

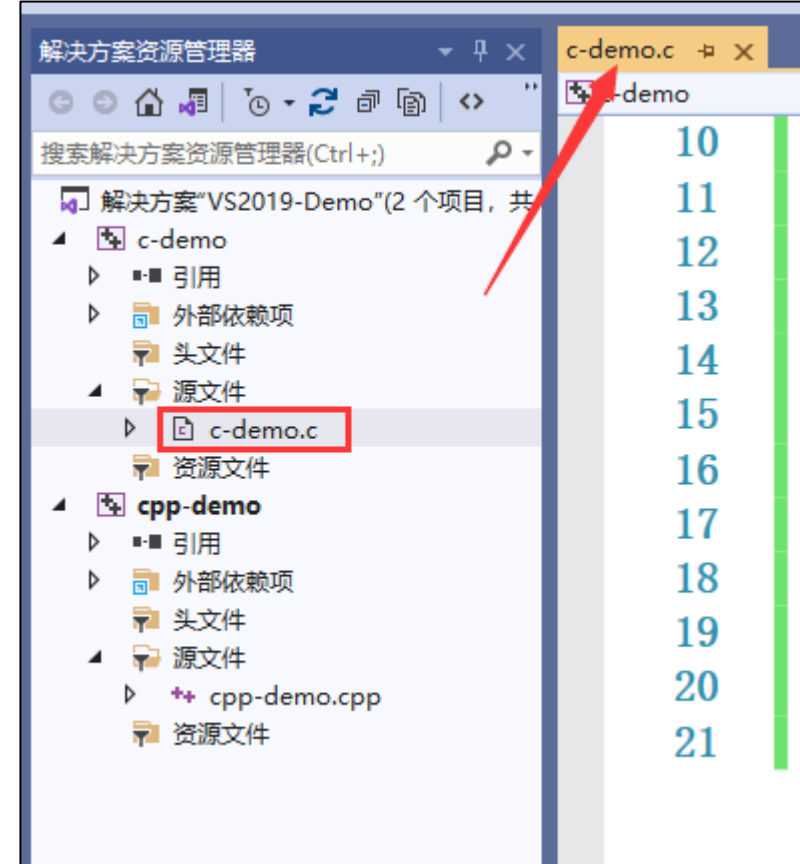


## § . 基础知识题 - C方式输入输出的格式化控制

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!!!

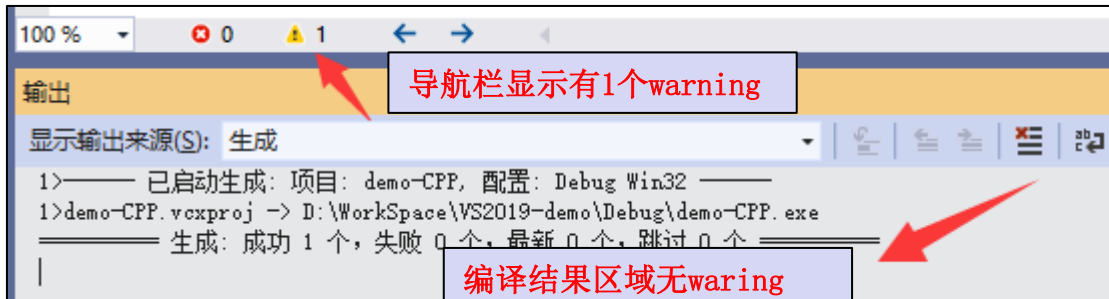


## §. 基础知识题 - C方式输入输出的格式化控制

★ 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 }
11
```



- 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

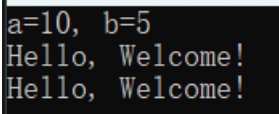
本页不用作答



# § . 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>	<pre>//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 #include "iostream" using namespace std;  int main() {     int a=10, b=5;     cout&lt;&lt;"a="&lt;&lt;a&lt;&lt;" , b="&lt;&lt;b&lt;&lt;endl;     cout&lt;&lt;"Hello, Welcome!"&lt;&lt;endl;     cout&lt;&lt;"Hello, Welcome!"&lt;&lt;endl;     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？ ！ 转义符在格式控制表列中的输出形式是：__字符____(字符/整数/转义符)</p>	

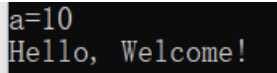
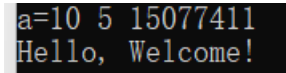




# § . 基础知识题 - C方式输入输出的格式化控制

## 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>运行结果:</p> 
<p>结论: 如果%d(格式符的数量) 小于 后面输出表列的数量, 则__忽略多出来的变量__</p>	<p>结论: 如果%d(格式符的数量) 大于 后面输出表列的数量, 则__ 输出已有的变量, 多出的会输出不可信值__</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:

```
a=10, b=5
a=10 b=5
a=10000
Hello
Hello
10 9 8 6 5
```

printf的返回值的含义是:  
返回输出的字符数



# §. 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

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:

```
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: 表示长整型整数，用于d, o, x, u前

附加控制符h的作用: 表示短整型整数，用于d, o, x, u前

★ 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



# § . 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
```

运行结果：

```
a=70000*
a=      70000*
a=70000  *
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：

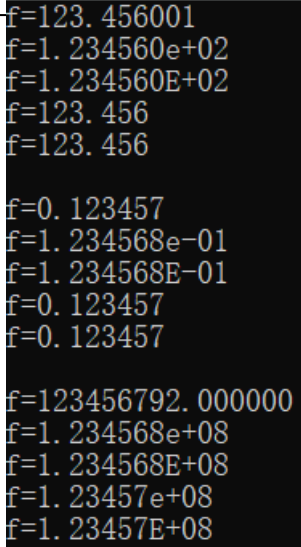
%ld : 以\_\_longlong类型的数据类型输出  
%10ld : 以\_\_longlong类型输出，总宽度\_10\_，右对齐  
%-10ld: 以\_\_longlong类型输出，总宽度\_10\_，左对齐  
%d : 以\_\_int\_\_类型的数据类型输出  
%10d : 以\_\_int\_\_类型输出，总宽度\_10\_，右对齐  
%-10d: 以int\_\_类型输出，总宽度\_10\_，左对齐  
%hd : 以\_\_short\_\_类型的数据类型输出  
%10hd : 以\_\_short\_\_类型输出，总宽度\_10\_，右对齐  
%-10hd: 以\_\_short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)

<pre>#include &lt;stdio.h&gt;  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; }</pre>	<div>运行结果:</div>  <p>参考printf的格式控制符和附加格式控制符，给出解释：</p> <p>%f：将浮点数以十进制的__小数__形式输出</p> <p>%e：将浮点数以十进制的__指数__形式输出</p> <p>%E：将浮点数以十进制的__指数__形式输出，</p> <p>%e和%E的区别是__指数e的大小写区别__</p> <p>%g/%G：输出形式为__选择输出宽度更短的形式输出__</p> <p>★ 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</p> <p>%g/%G：输出形式的差别为__指数e的大小写区别__</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

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;
}
```

运行结果:

```
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
```

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?

有区别，一个是输出float类型，一个是double类型

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?)





# §. 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", f);

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

```
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\_\_浮点数\_\_类型输出，总宽度\_\_10\_\_\_\_，  
小数点后\_\_2\_\_位，\_\_右\_\_对齐

%-10.2f: 以\_\_浮点数\_\_类型输出，总宽度\_\_10\_\_\_\_，  
小数点后\_\_2\_\_位，\_\_左\_\_对齐

%10.2e : 以\_\_浮点数\_\_类型输出，总宽度\_\_10\_\_\_\_，  
小数点后\_\_2\_\_\_\_位，\_\_右\_\_\_\_对齐

%-10.2e: 以\_\_浮点数\_\_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\_\_\_\_精确到小数点后n位\_\_\_\_

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：

```
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：

仍然会输出要求的位数但是超过精度后的值为不可信值

2、如果指定的总宽度小于有效位数的宽度，则：

按有效位数输出，截去多出的位数，并且四舍五入





## §. 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>

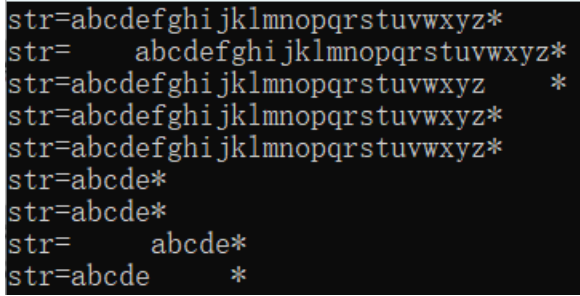
#define str "abcdefghijklmnopqrstuvwxyz"

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:



```
str=abcdefghijklmnopqrstuvwxyz*
str=      abcdefghi jklmnopqrstuvwxyz*
str=abcdefghijklmnopqrstuvwxyz *
str=abcdefghijklmnopqrstuvwxyz*
str=abcde*
str=abcde*
str=      abcde*
str=abcde *
```

参考printf的格式控制符和附加格式控制符，给出解释：

%s : 输出\_\_\_\_\_类型的数据

%30s : 输出\_\_\_\_\_类型的数据，总宽度\_\_\_\_\_,  
\_\_\_\_\_对齐

%-30s: 输出\_\_\_\_\_类型的数据，总宽度\_\_\_\_\_,  
\_\_\_\_\_对齐

如果指定的总宽度小于字符串的长度，则：

对%s而言，%m.n中n代表的位数是指\_\_\_\_\_



## §. 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>

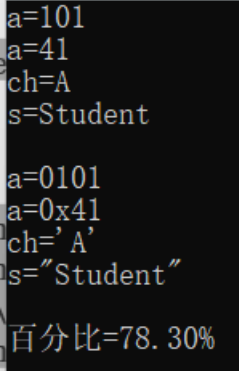
#define str "Student"
int main()
{
    int a = 65;
    printf("a=%o\n", a);
    printf("a=%x\n", a);
    printf("ch=%c\n", a);
    printf("s=%s\n\n", str);

    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:



- 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\_\_逗号  
后变量的输出，若需要前导字符、单双引号等，需要\_\_  
自行写出\_\_
- 2、输出字符‘%’的方法是：\_\_\_\_\_连续打两个  
%\_\_\_\_\_



# § . 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)

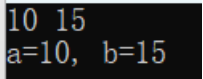
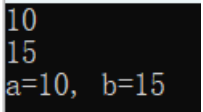
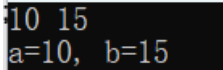
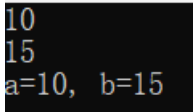
<pre>#define _CRT_SECURE_NO_WARNINGS #include &lt;stdio.h&gt;  int main() {     int a;     scanf("%d", a);     printf("a=%d\n", a);     return 0; }</pre>	<pre>#define _CRT_SECURE_NO_WARNINGS #include &lt;stdio.h&gt;  int main() {     int a = 0;     scanf("%d", a);     printf("a=%d\n", a);     return 0; }</pre>
<p>在VS中编译：</p> <div><div> 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"</div><div> C4700 使用了未初始化的局部变量"a"</div></div> <p>在Dev中编译：</p> <p>假设键盘输入为：10✓ (✓表示回车键，下同)</p> <p>则输出为：</p> <div>10</div>	<p>在VS中编译：</p> <p>假设键盘输入为：10✓</p> <p>则输出为：</p> <div></div> <p>在Dev中编译：</p> <p>假设键盘输入为：10✓</p> <p>则输出为：</p> <div>10</div> <p>结论：用scanf输入时，如果地址表列中直接跟变量名，则_____ (错误/正确)，其中VS的表现是__直接报错__，Dev的表现是__不会报错，但是后面的程序不会执行__</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)

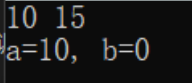
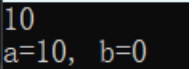
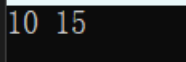
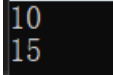
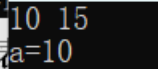
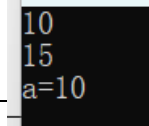
<pre>#define _CRT_SECURE_NO_WARNINGS #include &lt;stdio.h&gt;  int main() {     int a, b;     scanf("%d %d", &amp;a, &amp;b);     printf("a=%d, b=%d\n", a, b);     return 0; }</pre>	<pre>#define _CRT_SECURE_NO_WARNINGS #include &lt;stdio.h&gt;  int main() {     int a, b;     scanf("%d%d", &amp;a, &amp;b); //%d间无空格     printf("a=%d, b=%d\n", a, b);     return 0; }</pre>
<p>假设键盘输入为: <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>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>结论: 多个输入时, 格式控制符间是否有空格_不影响 (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>		
<p>假设键盘输入为: 10 15✓ 则输出为: </p> <p>假设键盘输入为: 10✓ 则输出为: </p> <p>结论: 当地址表列的个数多于格式控制符时, _将值赋给列表中前面的变量_____</p>	<p>VS: 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>Dev: 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: 当格式控制符的个数多个地址表列时____vs不会执行后面语句 Dev会将值输入给地址列表里有的地址____</p>		

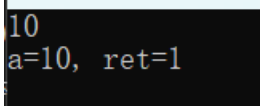
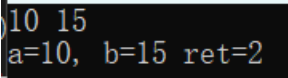




# § . 基础知识题 - C方式输入输出的格式化控制

## 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)

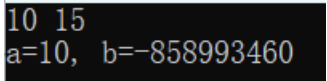
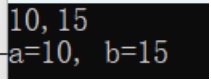
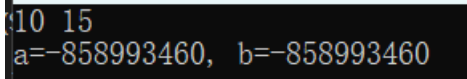
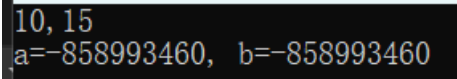
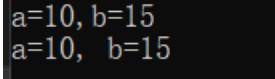
<pre>#define _CRT_SECURE_NO_WARNINGS #include &lt;stdio.h&gt;  int main() {     int a, ret;     ret = scanf("%d", &amp;a);     printf("a=%d, ret=%d\n", a, ret);     return 0; }</pre>	<pre>#define _CRT_SECURE_NO_WARNINGS #include &lt;stdio.h&gt;  int main() {     int a, b, ret;     ret = scanf("%d %d", &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__成功读取的数据个数__</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_____将控制字符用数据代替, 其他的字符保持原样输入_____</p>

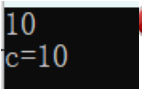
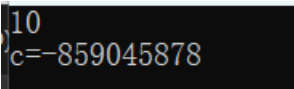

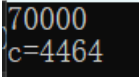




# § . 基础知识题 - C方式输入输出的格式化控制

## 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: 10✓ 则输出为: </p>	<p>假设键盘输入为: 10✓ 则输出为: </p>	<p>假设键盘输入为: 10✓ 则输出为: </p> <p>假设键盘输入为: 70000✓ 则输出为: </p>

结论:

1、附加格式控制符h的作用是\_\_\_\_输入短整型数, \_\_\_\_

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\_报错或者读取错误的数据\_\_\_\_\_

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!!!



## §. 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:

```
10 11 12
a=10, b=17, c=10
```

假设键盘输入为: 12 ab 76✓

则输出为:

```
12 ab 76
a=12, b=171, c=62
```

假设键盘输入为: 10 -11 +12✓

则输出为:

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

假设键盘输入为: 12 -ab +76✓

则输出为:

```
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:

```
10 11 12
a=10, b=17, c=10
```

假设键盘输入为: 12 ab 76✓

则输出为:

```
12 ab 76
a=12, b=171, c=62
```

假设键盘输入为: 10 -11 +12✓

则输出为:

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

假设键盘输入为: 12 -ab +76✓

则输出为:

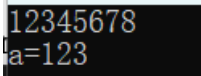
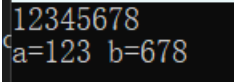
```
12 -ab +76
a=12, b=-171, c=62
```



# § . 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

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: 12345678✓ 则输出为: </p> <p>结论: %md中的m表示: 指定输入数据所占的宽度</p>	<p>假设键盘输入为: 12345678✓ 则输出为: </p> <p>结论: *md的*m表示: 跳过m个数据的宽度</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: 123✓ 则输出为: 123</p> <p>假设键盘输入为: 123 456✓ 则输出为: 123 456</p> <p>假设键盘输入为: 123a**✓ 则输出为: 123a** 123</p>	<p>假设键盘输入为: 123✓ 则输出为: 123 291</p> <p>假设键盘输入为: 123 456✓ 则输出为: 123 456 291</p> <p>假设键盘输入为: 123a**✓ 则输出为: 123a** 4666</p>	<p>假设键盘输入为: 123✓ 则输出为: 123 123</p> <p>假设键盘输入为: 123a**✓ 则输出为: 123a** 123</p> <p>假设键盘输入为: 12a**✓ 则输出为: 12a** 12</p>
<p>结论: scanf输入的终止条件是____空格____、 ____回车____、 ____到达宽度限制____和____不合法的字符____(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>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>注: 特别关注第4项的结果, 想想为什么? 宽度限制为3, 所以先读三个字符, 然后一个字符后遇到空格停止读取, 所以b为4</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\_\_\_\_\_读取双精度浮点数\_\_\_\_\_
- 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\_\_\_\_报错或者读取错误的\_\_\_\_数据\_\_\_\_\_
- 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>12.3456</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>1234.5678</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>12345678</u>✓ 则输出为: </p>
<p>结论:</p> <p>1、%mf/%mLf如果指定了宽度m, 则_____按宽度读取, 超出的部分不会写入_____</p> <p>2、%m.nf/%m.nLf如果指定了精度(小数点后的位数), 则_如果格式不符合则会写入错误的数据_____ (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)

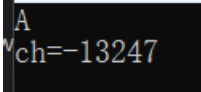
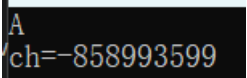
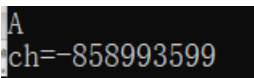
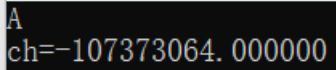
<pre>#define _CRT_SECURE_NO_WARNINGS #include &lt;stdio.h&gt; int main() {     char c1, c2;     scanf("%c %c", &amp;c1, &amp;c2);     printf("c1=%c c2=%c\n", c1, c2);     return 0; }</pre>	<pre>#define _CRT_SECURE_NO_WARNINGS #include &lt;stdio.h&gt; int main() {     char c1, c2;     scanf("%c%c", &amp;c1, &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: ABCD✓ 则输出为: ABCD c1=A c2=B</p> <p>假设键盘输入为: A BCD✓ 则输出为: A BCD c1=A c2=B</p> <p>假设键盘输入为: 'A' BCD✓ 则输出为: 'A' BCD c1=' c2=A</p> <p>假设键盘输入为: \n✓ 则输出为: \n c1=\ c2=n</p>	<p>假设键盘输入为: ABCD✓ 则输出为: ABCD c1=65 c2=66</p> <p>假设键盘输入为: A BCD✓ (特别关注此项的差异) 则输出为: A BCD c1=65 c2=32</p> <p>假设键盘输入为: 'A' BCD✓ 则输出为: 'A' BCD c1=39 c2=65</p> <p>假设键盘输入为: \n✓ 则输出为: \n c1=92 c2=110</p>
<p>结论:</p> <p>1、%c只读__1__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__特殊字符自身的ASCII码__ (特殊字符自身的ASCII码/特殊字符的转义含义)</p> <p>3、空格__不是__(是/不是)scanf中%c方式的有效输入，但必须注意__scan也可以读入空格__</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__非字符__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)  
则输出为:

Debug Error!  
Program: C:\Users\linchang\Desktop\Project1\Debug\Project2.exe  
Module: C:\Users\linchang\Desktop\Project1\Debug\Project2.exe  
File:  
Run-Time Check Failure #2 - Stack around the variable 's1' was corrupted.  
(Press Retry to debug the application)

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

则输出为:

Debug Error!  
Program: C:\Users\linchang\Desktop\Project1\Debug\Project2.exe  
Module: C:\Users\linchang\Desktop\Project1\Debug\Project2.exe  
File:  
Run-Time Check Failure #2 - Stack around the variable 's1' was corrupted.  
(Press Retry to debug the application)

结论:

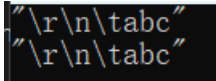
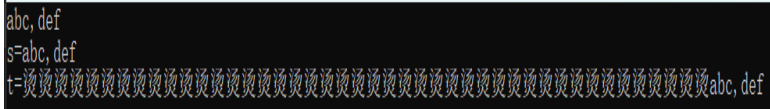
- 1、%s\_\_不能\_\_(能/不能)读入含空格的字符串
- 2、%s输入时，如果数组的大小为n，则最多输入\_\_n-1\_\_个字符



# § . 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)

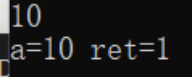
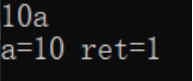
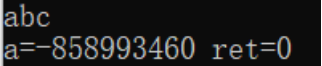
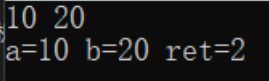
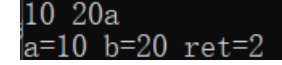
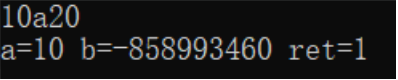
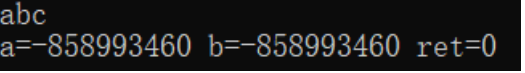
<pre>#define _CRT_SECURE_NO_WARNINGS #include &lt;stdio.h&gt; int main() {     char s[80];     scanf("%s", s);     printf("%s\n", s);     return 0; }</pre>	<pre>#define _CRT_SECURE_NO_WARNINGS #include &lt;stdio.h&gt; int main() {     char s[80], t[80];     scanf("%s,%s", s,t);     printf("s=%s\n", s);     printf("t=%s\n", t);     return 0; }</pre>
<p>假设键盘输入为: <u>"\r\n\tabc"</u>✓</p> <p>则输出为:</p>  <p>该字符串真正的内存存储为_11_个字节，这些字节的值 分别是_____“ \ r \ n \ t a b c ” 这11个所对应的ASCII码_____</p>	<p>假设键盘输入为: <u>abc,def</u>✓</p> <p>则输出为:</p>  <p>与2-E不同，“%s,%s”之间的逗号是____当做第一个字符串的有效字符____ (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__成功读取的数据的个数__</p>	