



# § . 基础知识题 – 输入输出部分

说明：C++中的格式控制很丰富，实现方法也有多种，下表列出的只是常用一部分，用于本次作业

控制符	作用
dec	设置整数为10进制
hex	设置整数为16进制
oct	设置整数为8进制
setbase(n)	设置整数为n进制 (n=8, 10, 16)
setfill(c)	设置填充字符，c可以是字符常量或字符变量
setprecision(n)	设置实数的精度为n位。在以一般十进制形式输出时，n代表有效数字。 在以fixed(固定小数位)形式和scientific(指数)形式输出时，n为小数位数
setw(n)	设置字段宽度为n
setiosflags(ios::fixed)	设置浮点数以固定的小数位数显示
setiosflags(ios::scientific)	设置浮点数以科学计数法（即指数形式）显示
setiosflags(ios::left)	输出数据左对齐
setiosflags(ios::right)	输出数据右对齐
setiosflags(ios::skipws)	忽略前导的空格
setiosflags(ios::uppercase)	在以科学计数法输出E和十六进制输出字母X时，以大写表示
setiosflags(ios::showpos)	输出正数时，给出“+”号
resetiosflags	终止已设置的输出格式状态，在括号中应指定内容



## §. 基础知识题 - 输入输出部分

### 8、在cout中使用格式化控制符

A. 进制前导符的使用：按要求自行构造测试程序，回答问题并将程序的运行结果截图贴上(允许多页)

```
#include <iostream>
#include <iomanip>

using namespace std;
int main()
{
    int a1 = ____; //常量为10进制表示正数
    cout << "dec:" << dec << a1 << endl;
    cout << "hex:" << hex << a1 << endl;
    cout << "oct:" << oct << a1 << endl;

    //构造a2: 常量为16进制表示, 且10进制输出形式为正数

    //构造a3: 常量为8进制表示, 且10进制输出形式为正数

    //构造a4: 常量为2进制表示, 且10进制输出形式为正数

    //构造a5: 常量为10进制表示负数

    //构造a6: 常量为16进制, 但10进制输出形式为负数

    //构造a7: 常量为8进制, 但10进制输出形式为负数

    //构造a8: 常量为2进制, 但10进制输出形式为负数

    return 0;
}
```

//允许将构造的程序直接贴图上来, 允许多页

自行构造若干组测试数据, 运行并截图

结论:

- 1、源程序中的整数, 有\_\_4\_\_种不同进制的表示形式
- 2、无论源程序中整型常量表示为何种进制, 它的机内存储均为\_\_2进制\_\_形式
- 3、如果想使数据输出时使用不同进制, 要加\_\_0b, 0, 0x\_\_等进制前导符
- 4、输出\_\_无\_\_(有/无)二进制前导符
- 5、只有\_\_10\_\_进制有负数形式输出;  
16进制输出负数时, 特征是\_\_前面是fff这样的形式开头\_\_;  
8进制输出负数时, 特征是\_\_前面是3777这样的形式开头\_\_



# a1

```
#include <iostream>
#include <iomanip>

using namespace std;

int main()
{
    int a1 = 18; //常量为10进制表示正数
    cout << "dec:" << dec << a1 << endl;
    cout << "hex:" << hex << a1 << endl;
    cout << "oct:" << oct << a1 << endl;
    return 0;
}
```

Microsoft Visual Studio 调试控制台

```
dec:18
hex:12
oct:22
```

# a2

```
#include <iomanip>
```

```
using namespace std;

int main()
```

```
int a2 = 0xa2; //常量为16进制表示，且10进制输出
cout << "dec:" << dec << a2 << endl;
cout << "hex:" << hex << a2 << endl;
cout << "oct:" << oct << a2 << endl;

return 0;
```

Microsoft Visual Studio 调试控制台

```
dec:162
hex:a2
oct:242
```



# a3

```
#include <iostream>
#include <iomanip>

using namespace std;

int main()
{
    int a3 = 0101; //常量为8进制表示, 且10进制输出形
    cout << "dec:" << dec << a3 << endl;
    cout << "hex:" << hex << a3 << endl;
    cout << "oct:" << oct << a3 << endl;
    return 0;
}
```

Microsoft Visual Studio 调试控制台

```
dec:65
hex:41
oct:101
```

# a4

```
using namespace std;
int main()
```

```
int a4 = 0b10010; //常量为2进制表示, 且10进制
cout << "dec:" << dec << a4 << endl;
cout << "hex:" << hex << a4 << endl;
cout << "oct:" << oct << a4 << endl;
return 0;
```

Microsoft Visual Studio 调试控制台

```
dec:18
hex:12
oct:22
```



# a5

```
#include <iomanip>
```

```
using namespace std;
```

```
int main()
```

```
int a5 = -123; //常量为10进制表示负数
```

```
cout << "dec:" << dec << a5 << endl;
```

```
cout << "hex:" << hex << a5 << endl;
```

```
cout << "oct:" << oct << a5 << endl;
```

```
return 0;
```

Microsoft Visual Studio 调试控制台

```
dec:-123
hex:ffffff85
oct:37777777605
```

# a6

```
int a6 = 0xffffffff; //常量为16进制，但10进制输出
```

```
cout << "dec:" << dec << a6 << endl;
```

```
cout << "hex:" << hex << a6 << endl;
```

```
cout << "oct:" << oct << a6 << endl;
```

```
return 0;
```

Microsoft Visual Studio 调试控制台

```
dec:-1
hex:ffffffff
oct:37777777777
```



# a7

```
int main()
{
    int a7 = 037777777766; //常量为2进制,
    cout << "dec:" << dec << a7 << endl;
    cout << "hex:" << hex << a7 << endl;
    cout << "oct:" << oct << a7 << endl;
    return 0;
}
```

Microsoft Visual Studio 调试控制台

```
dec:-10
hex:ffffff6
oct:37777777766
```

# a8

```
int main()
{
    int a8 = 0b11111111111111111111111111111111010;
    cout << "dec:" << dec << a8 << endl;
    cout << "hex:" << hex << a8 << endl;
    cout << "oct:" << oct << a8 << endl;
    return 0;
}
```

Microsoft Visual Studio 调试控制台

```
dec:-6
hex:fffffffa
oct:3777777772
```



## §. 基础知识题 – 输入输出部分

### 8、在cout中使用格式化控制符

B. 进制前导符的连续使用：回答问题并将程序的运行结果截图贴上

```
#include <iostream>
#include <iomanip>

using namespace std;
int main()
{
    int a = 10;

    cout << a << ' ' << a+1 << ' ' << a+2 << endl;
    cout << hex;
    cout << a << ' ' << a+1 << ' ' << a+2 << endl;
    cout << oct;
    cout << a << ' ' << a+1 << ' ' << a+2 << endl;
    cout << dec;
    cout << a << ' ' << a+1 << ' ' << a+2 << endl;

    return 0;
}
```

```
10 11 12
a b c
12 13 14
10 11 12
```

结论：

dec/hex/oct等进制前导符设置后，对后面的\_\_所有\_\_(仅一个/所有)数据有效，直到用另一个控制符去改变为止



# §. 基础知识题 - 输入输出部分

8、在cout中使用格式化控制符

C. setbase的使用：同8. A/8. B的形式，按要求自行构造测试程序，回答问题并将程序的运行结果截图贴上(允许多页)

<pre>#include &lt;iostream&gt; #include &lt;iomanip&gt; using namespace std; int main() {     int a =10; //常量为10进制表示     cout &lt;&lt;setbase(8)&lt;&lt;a &lt;&lt;endl;     cout &lt;&lt;setbase(10)&lt;&lt; a&lt;&lt;endl;     cout &lt;&lt;setbase(16)&lt;&lt;a &lt;&lt;endl;     //构造a2: 常量为16进制表示     //构造a3: 常量为字符型     //构造a4: 常量为浮点型     //构造a5: setbase里面放入其他数     return 0; }</pre> <p>//允许将构造的程序直接贴图上来，允许多页</p>	<p>自行构造若干组测试数据，运行并截图</p> <p>结论：</p> <p>1、setbase中允许的合法值有__8， 10， 16_____</p> <p>2、当setbase中出现非法值时，处理方法是__统一按照10进制处理__</p> <p>3、setbase设置后，对后面的__所有____(仅一个/所有)数据有效，直到用另一个setbase去改变为止</p>
---	---





# a1

```
int a = 10; //常量为10进制表示正数
cout << setbase(10) << a << endl;
cout << setbase(8) << a << endl;
cout << setbase(16) << a << endl;
return 0;
```

Microsoft Visual Studio 调试控制台

```
10
12
a
```

# a2

```
int a2 = 0xa2; //常量为16进制表示, 且10
cout << setbase(10) << a2 << endl;
cout << setbase(8) << a2 << endl;
cout << setbase(16) << a2 << endl;
return 0;
```

Microsoft Visual Studio 调试控制台

```
162
242
a2
```



a3

```
char a2 = 'A'; //常量为字符型
cout << setbase(10) << a2 << endl;
cout << setbase(8) << a2 << endl;
cout << setbase(16) << a2 << endl;
return 0;
```



a4

```
float a2 = 123.456; //常量为浮点型
cout << setbase(10) << a2 << endl;
cout << setbase(8) << a2 << endl;
cout << setbase(16) << a2 << endl;
return 0;
```





# a5

```
int a5 =10;  
cout <<setbase(15)<<a5 <<endl;  
cout <<setbase(3)<< a5 <<endl;  
cout <<setbase(6)<<a5 <<endl;  
cout <<setbase(12) << a5 << endl;  
return 0;
```

Microsoft Visual Studio 调试控制台

```
10  
10  
10  
10
```



## §. 基础知识题 - 输入输出部分

### 8、在cout中使用格式化控制符

#### C. setbase的使用:

```
#include <iostream>
#include <iomanip>

using namespace std;
int main()
{
    int a = 10;
    cout << setbase(10) << a << ' ' << a + 1 << ' ' << a + 2 << endl;
    cout << setbase(8) << a << ' ' << a + 1 << ' ' << a + 2 << endl;
    cout << setbase(16) << a << ' ' << a + 1 << ' ' << a + 2 << endl;
    return 0;
}
```

```
10 11 12
12 13 14
a b c
```

#### 结论:

setbase设置后, 对后面的\_\_所有\_\_(仅一个/所有)数据有效, 直到用另一个控制符去改变为止

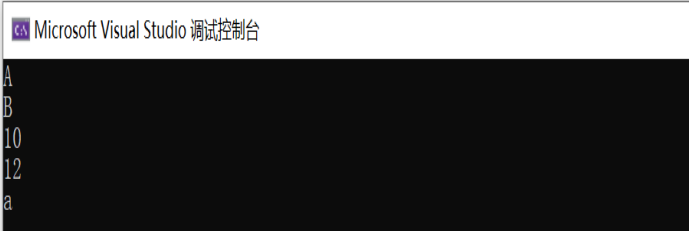


## §. 基础知识题 – 输入输出部分

### 8、在cout中使用格式化控制符

D. `ios::uppercase`的使用：按要求自行构造测试程序，能对比看出用和不用的差别即可

```
#include <iostream>
#include <iomanip>
using namespace std;
int main()
{
    int a = 10;
    cout << setiosflags(ios::uppercase) << hex << a << endl;
    cout << a + 1 << endl;
    cout << dec << a << endl;
    cout << oct << a << endl;
    cout << resetiosflags(ios::uppercase) << hex << a << endl;
    return 0;
}
```



测试程序中的数据类型为int，自行构造若干组测试数据，运行并截图

结论：

1、uppercase和\_16\_进制一起使用才能看出效果

2、uppercase设置后，对后面的\_\_所有\_\_\_\_(仅一个/所有)数据有效

3、同一个程序中，设置完uppercase，如果想恢复小写，具体的做法是\_\_设置resetiosflags(ios::uppercase)\_\_  
(本小问如果不会，先不要问，先往后做，看后面的题目是否有相似问题可以启发你)

//允许将构造的程序直接贴图上来

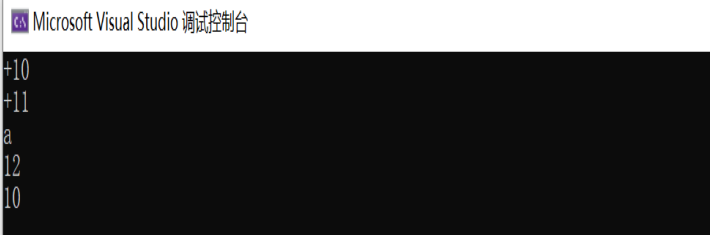


# § . 基础知识题 - 输入输出部分

## 8、在cout中使用格式化控制符

E. ios::showpos的使用：按要求自行构造测试程序，能对比看出用和不用的差别即可

```
#include <iostream>
#include <iomanip>
using namespace std;
int main()
{
    int a = 10;
    cout << setiosflags(ios::showpos) << dec << a << endl;
    cout << a + 1 << endl;
    cout << hex << a << endl;
    cout << oct << a << endl;
    cout << resetiosflags(ios::showpos) << dec << a << endl;
    return 0;
}
```



//允许将构造的程序直接贴图上来

测试程序中的数据类型为int，自行构造若干组测试数据，运行并截图

结论：

- 1、 showpos和\_10\_\_进制一起使用才能看出效果
- 2、 showpos设置后，对后面的\_\_所有\_\_\_\_(仅一个/所有)数据有效
- 3、 同一个程序中，设置完showpos，如果想取消，具体的做法是\_\_resetiosflags(ios::showpos)\_\_\_\_\_  
(本小问如果不会，先不要问，先往后做，看后面的题目是否有相似问题可以启发你)



## §. 基础知识题 – 输入输出部分

### 8、在cout中使用格式化控制符

#### F. setprecision的使用 – 单独使用 – (1)

```
#include <iostream>
#include <iomanip>
using namespace std;
int main()
{
    float f1 = 1234.5678F, f2 = 8765.4321F;
    /* 第1组：不设或非法 */
    cout << f1 << ' ' << f2 << endl;
    cout << setprecision(0) << f1 << ' ' << f2 << endl;
    /* 第2组：小于等于整数位数 */
    cout << endl;
    cout << setprecision(1) << f1 << ' ' << f2 << endl;
    cout << setprecision(2) << f1 << ' ' << f2 << endl;
    cout << setprecision(3) << f1 << ' ' << f2 << endl;
    cout << setprecision(4) << f1 << ' ' << f2 << endl;
    /* 第3组：大于整数位数，但小于等于float型有效数字 */
    cout << endl;
    cout << setprecision(5) << f1 << ' ' << f2 << endl;
    cout << setprecision(6) << f1 << ' ' << f2 << endl;
    /* 第4组：大于float型有效数字 */
    cout << endl;
    cout << setprecision(7) << f1 << ' ' << f2 << endl;
    cout << setprecision(8) << f1 << ' ' << f2 << endl;
    cout << setprecision(9) << f1 << ' ' << f2 << endl;
    cout << setprecision(10) << f1 << ' ' << f2 << endl;
    cout << setprecision(20) << f1 << ' ' << f2 << endl;
    cout << setprecision(50) << f1 << ' ' << f2 << endl;

    return 0;
}
```

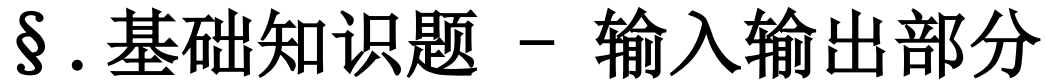
#### 本例贴图

```
1234.57 8765.43
1234.57 8765.43

1e+03 9e+03
1.2e+03 8.8e+03
1.23e+03 8.77e+03
1235 8765

1234.6 8765.4
1234.57 8765.43
1234.568 8765.432

1234.5677 8765.4316
1234.56775 8765.43164
1234.567749 8765.431641
1234.5677490234375 8765.431640625
1234.5677490234375 8765.431640625
```



### F. setprecision的使用 - 单独使用 - (2)

```
f1 = 1234567890123456789.0F;  
f2 = 9876543210987654321.0F;
```

```
return 0;
```

F:\C++练习\各种习题练习\Debug\各种习题练习.exe  
要在调试停止时自动关闭控制台，请启用“工具”  
按任意键关闭此窗口. . .





## §. 基础知识题 - 输入输出部分

### 8、在cout中使用格式化控制符

#### F. setprecision的使用 - 单独使用 - (3)

```
#include <iostream>
#include <iomanip>
using namespace std;
int main()
{
    float f1 = 0.12345678F, f2 = 0.87654321F;

    /* 第1组: 不设或非法 */
    cout << f1 << ' ' << f2 << endl;
    cout << setprecision(0) << f1 << ' ' << f2 << endl;
    /* 第2组: 小于等于float型有效数字 */
    cout << endl;
    cout << setprecision(1) << f1 << ' ' << f2 << endl;
    cout << setprecision(2) << f1 << ' ' << f2 << endl;
    cout << setprecision(5) << f1 << ' ' << f2 << endl;
    cout << setprecision(6) << f1 << ' ' << f2 << endl;
    /* 第3组: 大于float型有效数字, 但小于等于小数位数 */
    cout << endl;
    cout << setprecision(7) << f1 << ' ' << f2 << endl;
    cout << setprecision(8) << f1 << ' ' << f2 << endl;
    /* 第4组: 大于小数位数 */
    cout << endl;
    cout << setprecision(9) << f1 << ' ' << f2 << endl;
    cout << setprecision(10) << f1 << ' ' << f2 << endl;
    cout << setprecision(20) << f1 << ' ' << f2 << endl;
    cout << setprecision(40) << f1 << ' ' << f2 << endl;
    cout << setprecision(50) << f1 << ' ' << f2 << endl;

    return 0;
}
```

数据换为:

f1 = 0.12345678F;

f2 = 0.87654321F;

0.123457 0.876543  
0.123457 0.876543

0.1 0.9  
0.12 0.88  
0.12346 0.87654  
0.123457 0.876543

0.1234568 0.8765432  
0.12345678 0.87654322

0.123456784 0.876543224  
0.1234567836 0.8765432239  
0.12345678359270095825 0.87654322385787963867  
0.123456783592700958251953125 0.876543223857879638671875  
0.123456783592700958251953125 0.876543223857879638671875

F:\C++练习\各种习题练习\Debug\各种习题练习.exe (进程 1640)  
要在调试停止时自动关闭控制台, 请启用“工具”->“选项”->  
按任意键关闭此窗口. . .



## §. 基础知识题 – 输入输出部分

### 8、在cout中使用格式化控制符

#### F. setprecision的使用 – 单独使用 – 总结

**重要结论：** setprecision指定输出位数后，系统会按指定位数输出，即使指定位数超过数据的有效位数（即：输出数据的某位开始是不可信的，但依然会输出）

1、给出setprecision单独使用时的显示规律总结（如果数据不够，可以再自己构造测试数据）

float的6位有效位数以内的数按照四舍五入保留，7位数以后到某一个上限是随机出现的数字，并不根据本身输入的数字来确定数字，而这个上限的位数还是不确定的，超过这个上限的位数，超过这个精度以后的位数统一不算，长度固定了。

2、将8. F-(1)~(3)中的数据类型换为double型（有效位数为15位），自行构造测试数据，验证总结出的float型数据的显示规律是否同样适用于double型（如果适用，不用贴图，如果不适用，贴对应代码及运行截图）

double的数据同样适用，只不过有效位数变成15位，其余规律与float一致。



## §. 基础知识题 – 输入输出部分

### 8、在cout中使用格式化控制符

#### G. setprecision的使用 – 和ios::fixed一起 – (1)

```
#include <iostream>
#include <iomanip>

using namespace std;
int main()
{
    float f1 = 1234.5678F, f2 = 8765.4321F;

    /* 第1组: 不设precision */
    cout << f1 << ' ' << f2 << endl;
    cout << setiosflags(ios::fixed) << f1 << ' ' << f2 << endl;

    /* 第2组: 设置precision */
    cout << endl;
    cout << setprecision(1) << f1 << ' ' << f2 << endl;
    cout << setprecision(4) << f1 << ' ' << f2 << endl;
    cout << setprecision(7) << f1 << ' ' << f2 << endl;
    cout << setprecision(10) << f1 << ' ' << f2 << endl;
    cout << setprecision(20) << f1 << ' ' << f2 << endl;

    return 0;
}
```

贴图:

```
1234.57 8765.43
1234.567749 8765.431641

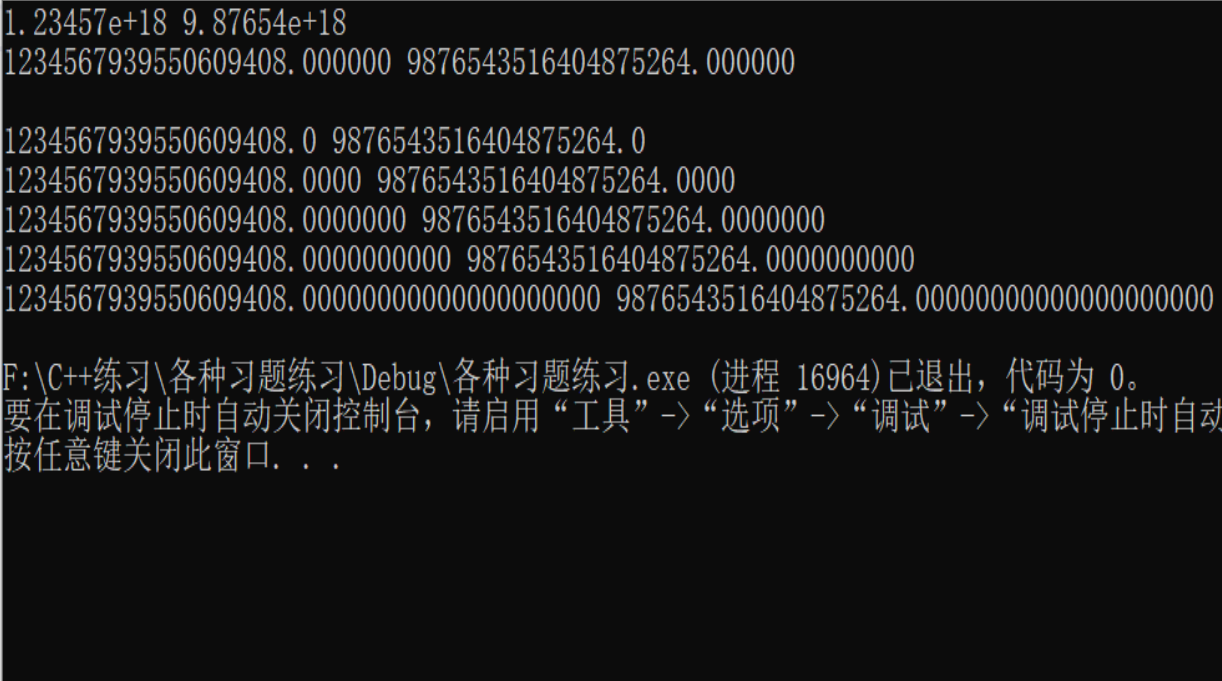
1234.6 8765.4
1234.5677 8765.4316
1234.5677490 8765.4316406
1234.5677490234 8765.4316406250
1234.56774902343750000000 8765.43164062500000000000
```



# § . 基础知识题 - 输入输出部分

## 8、在cout中使用格式化控制符

### G.setprecision的使用 - 和ios::fixed一起 - (2)

<pre>#include &lt;iostream&gt; #include &lt;iomanip&gt;  using namespace std; int main() {     float f1 = 1234567890123456789.0F, f2 = 9876543210987654321.0F;     /* 第1组: 不设precision */     cout &lt;&lt; f1 &lt;&lt; ' ' &lt;&lt; f2 &lt;&lt; endl;     cout &lt;&lt; setiosflags(ios::fixed) &lt;&lt; f1 &lt;&lt; ' ' &lt;&lt; f2 &lt;&lt; endl;      /* 第2组: 设置precision */     cout &lt;&lt; endl;     cout &lt;&lt; setprecision(1) &lt;&lt; f1 &lt;&lt; ' ' &lt;&lt; f2 &lt;&lt; endl;     cout &lt;&lt; setprecision(4) &lt;&lt; f1 &lt;&lt; ' ' &lt;&lt; f2 &lt;&lt; endl;     cout &lt;&lt; setprecision(7) &lt;&lt; f1 &lt;&lt; ' ' &lt;&lt; f2 &lt;&lt; endl;     cout &lt;&lt; setprecision(10) &lt;&lt; f1 &lt;&lt; ' ' &lt;&lt; f2 &lt;&lt; endl;     cout &lt;&lt; setprecision(20) &lt;&lt; f1 &lt;&lt; ' ' &lt;&lt; f2 &lt;&lt; endl;      return 0; }</pre>	<p>数据换为:</p> <p>f1 = 1234567890123456789.0F; f2 = 9876543210987654321.0F;</p> <p>贴图:</p>  <p>F:\C++练习\各种习题练习\Debug\各种习题练习.exe (进程 16964) 已退出, 代码为 0。 要在调试停止时自动关闭控制台, 请启用“工具”-&gt;“选项”-&gt;“调试”-&gt;“调试停止时自动 按任意键关闭此窗口. . .</p>
---	--



## §. 基础知识题 - 输入输出部分

### 8、在cout中使用格式化控制符

#### G.setprecision的使用 - 和ios::fixed一起 - (3)

```
#include <iostream>
#include <iomanip>

using namespace std;
int main()
{
    float f1 = 0.12345678F, f2 = 0.87654321F;
    /* 第1组: 不设precision */
    cout << f1 << ' ' << f2 << endl;
    cout << setiosflags(ios::fixed) << f1 << ' ' << f2 << endl;

    /* 第2组: 设置precision */
    cout << endl;
    cout << setprecision(1) << f1 << ' ' << f2 << endl;
    cout << setprecision(4) << f1 << ' ' << f2 << endl;
    cout << setprecision(7) << f1 << ' ' << f2 << endl;
    cout << setprecision(10) << f1 << ' ' << f2 << endl;
    cout << setprecision(20) << f1 << ' ' << f2 << endl;

    return 0;
}
```

数据换为:

f1 = 0.12345678F;

f2 = 0.87654321F;

贴图:

```
0.123457 0.876543
0.123457 0.876543
```

```
0.1 0.9
0.1235 0.8765
0.1234568 0.8765432
0.1234567836 0.8765432239
0.12345678359270095825 0.87654322385787963867
```

F:\C++练习\各种习题练习\Debug\各种习题练习.exe  
要在调试停止时自动关闭控制台, 请启用“工具”->  
按任意键关闭此窗口. . .



## §. 基础知识题 – 输入输出部分

### 8、在cout中使用格式化控制符

#### G.setprecision的使用 – 和ios::fixed一起 – 总结

1、给出setprecision+ios::fixed使用时的显示规律总结（如果数据不够，可以再自己构造测试数据）

系统默认输出的有效位数为6位，setiosflags(ios::fixed)是固定了小数位数默认为6位，后面如果再加setprecision()，那么改变的是小数位数，小数位为括号里的数字，但是因为float的精度是6位，那么只要超过范围的数字都是随机或者说是无效的。

2、将8. G-(1)~(3)中的数据类型换为double型（有效位数为15位），自行构造测试数据，验证总结出的float型数据的显示规律是否同样适用于double型（如果适用，不用贴图，如果不适用，贴对应代码及运行截图）

同样适用，只不过是double变成了15位有效数字，但是setiosflags(ios::fixed)还是固定了小数位数默认为6位，也就是说虽然double是15位，但是因为固定了6位，所以只能输出6位，但是可以通过改变setprecision()来控制输出的位数。





## §. 基础知识题 – 输入输出部分

### 8、在cout中使用格式化控制符

#### H. setprecision的使用 – 和ios::scientific一起 – (1)

```
#include <iostream>
#include <iomanip>

using namespace std;
int main()
{
    float f1 = 1234.5678F, f2 = 8765.4321F;

    /* 第1组: 不设precision */
    cout << f1 << ' ' << f2 << endl;
    cout << setiosflags(ios::scientific) << f1 << ' ' << f2 << endl;

    /* 第2组: 设置precision */
    cout << endl;
    cout << setprecision(1) << f1 << ' ' << f2 << endl;
    cout << setprecision(4) << f1 << ' ' << f2 << endl;
    cout << setprecision(7) << f1 << ' ' << f2 << endl;
    cout << setprecision(10) << f1 << ' ' << f2 << endl;
    cout << setprecision(20) << f1 << ' ' << f2 << endl;

    return 0;
}
```

贴图:

```
1234.57 8765.43
1.234568e+03 8.765432e+03

1.2e+03 8.8e+03
1.2346e+03 8.7654e+03
1.2345677e+03 8.7654316e+03
1.2345677490e+03 8.7654316406e+03
1.23456774902343750000e+03 8.76543164062500000000e+03
```



## §. 基础知识题 – 输入输出部分

### 8、在cout中使用格式化控制符

#### H. setprecision的使用 – 和ios::scientific一起 – (2)

```
#include <iostream>
#include <iomanip>

using namespace std;
int main()
{
    float f1 = 1234567890123456789.0F, f2 = 9876543210987654321.0F;
    /* 第1组: 不设precision */
    cout << f1 << ' ' << f2 << endl;
    cout << setiosflags(ios::scientific) << f1 << ' ' << f2 << endl;
    /* 第2组: 设置precision */
    cout << endl;
    cout << setprecision(1) << f1 << ' ' << f2 << endl;
    cout << setprecision(4) << f1 << ' ' << f2 << endl;
    cout << setprecision(7) << f1 << ' ' << f2 << endl;
    cout << setprecision(10) << f1 << ' ' << f2 << endl;
    cout << setprecision(20) << f1 << ' ' << f2 << endl;
}
```

数据换为:

f1 = 1234567890123456789.0F;

f2 = 9876543210987654321.0F;

贴图:

```
1.23457e+18 9.87654e+18
1.234568e+18 9.876544e+18

1.2e+18 9.9e+18
1.2346e+18 9.8765e+18
1.2345679e+18 9.8765435e+18
1.2345679396e+18 9.8765435164e+18
1.23456793955060940800e+18 9.87654351640487526400e+18
```

F:\C++练习\各种习题练习\Debug\各种习题练习.exe (进程 1111)  
要在调试停止时自动关闭控制台, 请启用“工具”->“选项”->“调试”->“在调试停止时关闭控制台”  
按任意键关闭此窗口. . .





## §. 基础知识题 - 输入输出部分

### 8、在cout中使用格式化控制符

#### H. setprecision的使用 - 和ios::scientific一起 - (3)

```
#include <iostream>
#include <iomanip>

using namespace std;
int main()
{
    float f1 = 0.12345678F, f2 = 0.87654321F;
    /* 第1组: 不设precision */
    cout << f1 << ' ' << f2 << endl;
    cout << setiosflags(ios::scientific) << f1 << ' ' << f2 << endl;
    /* 第2组: 设置precision */
    cout << endl;
    cout << setprecision(1) << f1 << ' ' << f2 << endl;
    cout << setprecision(4) << f1 << ' ' << f2 << endl;
    cout << setprecision(7) << f1 << ' ' << f2 << endl;
    cout << setprecision(10) << f1 << ' ' << f2 << endl;
    cout << setprecision(20) << f1 << ' ' << f2 << endl;

    return 0;
}
```

数据换为:

f1 = 0.12345678F;

f2 = 0.87654321F;

贴图:

```
0.123457 0.876543
1.234568e-01 8.765432e-01

1.2e-01 8.8e-01
1.2346e-01 8.7654e-01
1.2345678e-01 8.7654322e-01
1.2345678359e-01 8.7654322386e-01
1.23456783592700958252e-01 8.76543223857879638672e-01
```

F:\C++练习\各种习题练习\Debug\各种习题练习.exe (进程 3  
要在调试停止时自动关闭控制台, 请启用“工具”->“选项”  
按任意键关闭此窗口. . .



## §. 基础知识题 – 输入输出部分

### 8、在cout中使用格式化控制符

#### H. setprecision的使用 – 和ios::scientific一起 – 总结

1、给出setprecision+ios::scientific使用时的显示规律总结（如果数据不够，可以再自己构造测试数据）

系统默认输出的有效位数为6位，setiosflags(ios::scientific)是固定了科学计数法里的小数位数默认为6位，后面如果再加setprecision()，那么改变的是小数位数，小数位为括号里的数字，但是因为float的精度是6位，那么只要超过范围的数字都是随机或者说是无效的。

2、将8.H-(1)~(3)中的数据类型换为double型（有效位数为15位），自行构造测试数据，验证总结出的float型数据的显示规律是否同样适用于double型（如果适用，不用贴图，如果不适用，贴对应代码及运行截图）

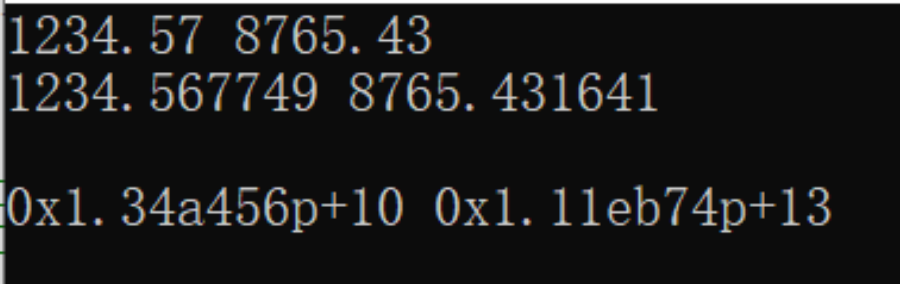
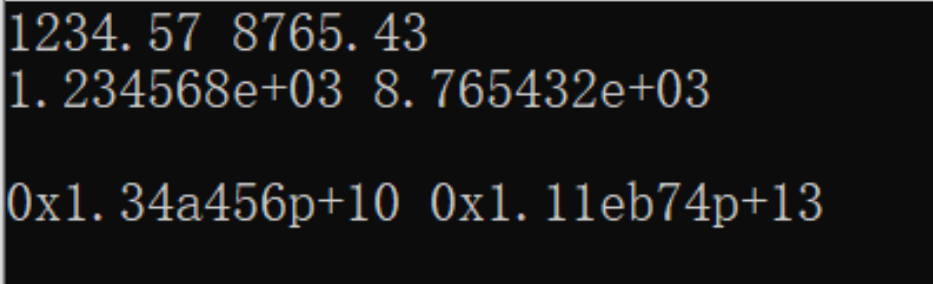
同样适用，只不过是double变成了15位有效数字，但是setiosflags(ios::scientific)还是固定了小数位数默认为6位，也就是说虽然double是15位，但是因为固定了6位，所以只能输出6位，但是可以通过改变setprecision()来控制输出的位数。



# §. 基础知识题 – 输入输出部分

## 8、在cout中使用格式化控制符

### I. ios::fixed和ios::scientific的混合使用 – 错误用法

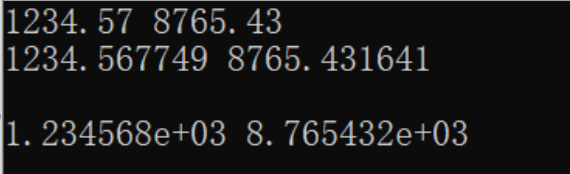
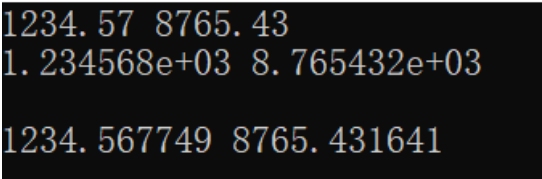
<pre>#include &lt;iostream&gt; #include &lt;iomanip&gt;  using namespace std; int main() {     float f1 = 1234.5678F, f2 = 8765.4321F;      /* 第1组 */     cout &lt;&lt; f1 &lt;&lt; ' ' &lt;&lt; f2 &lt;&lt; endl;     cout &lt;&lt; setiosflags(ios::fixed) &lt;&lt; f1 &lt;&lt; ' ' &lt;&lt; f2 &lt;&lt; endl;      /* 第2组 */     cout &lt;&lt; endl;     cout &lt;&lt; setiosflags(ios::scientific) &lt;&lt; f1 &lt;&lt; ' ' &lt;&lt; f2 &lt;&lt; endl;      return 0; }</pre>	<pre>#include &lt;iostream&gt; #include &lt;iomanip&gt;  using namespace std; int main() {     float f1 = 1234.5678F, f2 = 8765.4321F;      /* 第1组 */     cout &lt;&lt; f1 &lt;&lt; ' ' &lt;&lt; f2 &lt;&lt; endl;     cout &lt;&lt; setiosflags(ios::scientific) &lt;&lt; f1 &lt;&lt; ' ' &lt;&lt; f2 &lt;&lt; endl;      /* 第2组 */     cout &lt;&lt; endl;     cout &lt;&lt; setiosflags(ios::fixed) &lt;&lt; f1 &lt;&lt; ' ' &lt;&lt; f2 &lt;&lt; endl;      return 0; }</pre>
<p>运行截图:</p> 	<p>运行截图:</p> 



# § . 基础知识题 – 输入输出部分

## 8、在cout中使用格式化控制符

I. ios::fixed和ios::scientific的混合使用 – 在上一页的基础上将程序改正确，并给出截图

<pre>#include &lt;iostream&gt; #include &lt;iomanip&gt;  using namespace std; int main() {     float f1 = 1234.5678F, f2 = 8765.4321F;      /* 第1组 */     cout &lt;&lt; f1 &lt;&lt; ' ' &lt;&lt; f2 &lt;&lt; endl;     cout &lt;&lt; setiosflags(ios::fixed) &lt;&lt; f1 &lt;&lt; ' ' &lt;&lt; f2 &lt;&lt; endl;     cout&lt;&lt;resetiosflags(ios::fixed);     /* 第2组 */     cout &lt;&lt; endl;     cout &lt;&lt; setiosflags(ios::scientific) &lt;&lt; f1 &lt;&lt; ' ' &lt;&lt; f2 &lt;&lt; endl;      return 0; }</pre>	<pre>#include &lt;iostream&gt; #include &lt;iomanip&gt;  using namespace std; int main() {     float f1 = 1234.5678F, f2 = 8765.4321F;      /* 第1组 */     cout &lt;&lt; f1 &lt;&lt; ' ' &lt;&lt; f2 &lt;&lt; endl;     cout &lt;&lt; setiosflags(ios::scientific) &lt;&lt; f1 &lt;&lt; ' ' &lt;&lt; f2 &lt;&lt; endl;     cout&lt;&lt;resetiosflags(ios::scientific);     /* 第2组 */     cout &lt;&lt; endl;     cout &lt;&lt; setiosflags(ios::fixed) &lt;&lt; f1 &lt;&lt; ' ' &lt;&lt; f2 &lt;&lt; endl;      return 0; }</pre>
<p>运行截图:</p> 	<p>运行截图:</p> 
<p>结论: 如果想要在一个程序中同时显示fixed和scientific形式, 需要在两者之间加入一句: __cout&lt;&lt;resetiosflags(ios::fixed);或者cout&lt;&lt;resetiosflags(ios::fixed)____</p>	



# §. 基础知识题 - 输入输出部分

## 8、在cout中使用格式化控制符 J. setw的基本使用 - (1)

```
#include <iostream>
#include <iomanip>

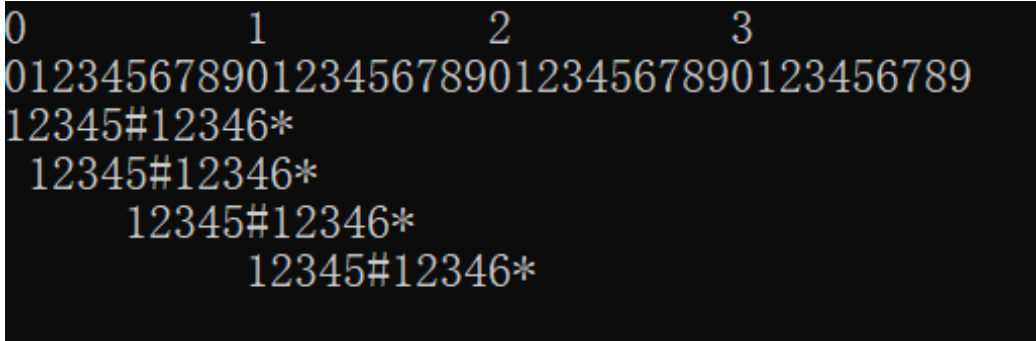
using namespace std;
int main()
{
    int a = 12345;

    cout << "0      1      2      3" << endl;
    cout << "0123456789012345678901234567890123456789" << endl;

    cout << setw(3) << a << '#' << a + 1 << '*' << endl;
    cout << setw(6) << a << '#' << a + 1 << '*' << endl;
    cout << setw(10) << a << '#' << a + 1 << '*' << endl;
    cout << setw(15) << a << '#' << a + 1 << '*' << endl;

    return 0;
}
```

运行截图：



- 结论：
- 1、setw指定的宽度是总宽度，当总宽度大于数据宽度时，显示规律为多余的位置补充空格，空格在数据前面\_；  
当总宽度小于数据宽度时，显示规律为 显示的是数据的宽度，并且无空格\_
  - 2、setw的设置后，对后面的\_\_仅一个\_\_\_\_(仅一个/所有)数据有效
  - 3、程序最前面两行的输出，目的是什么？ 为了下面的数据寻找位数
  - 4、每行输出的最后一个\*，目的是什么？ 观察右侧的相差位数，便于比较



## §. 基础知识题 - 输入输出部分

### 8、在cout中使用格式化控制符 J. setw的基本使用 - (2)

```
#include <iostream>
#include <iomanip>

using namespace std;
int main()
{
    double a = 0.123456789012345;

    cout << "0          1          2          3" << endl;
    cout << "0123456789012345678901234567890123456789" << endl;

    cout << setw(6) << a << '*' << endl;
    cout << setw(9) << a << '*' << endl;
    cout << setw(15) << a << '*' << endl;
    cout << setw(30) << a << '*' << endl;

    return 0;
}
```

运行截图:

```
0          1          2          3
0123456789012345678901234567890123456789
0.123457*
0.123457*
0.123457*
0.123457*
```

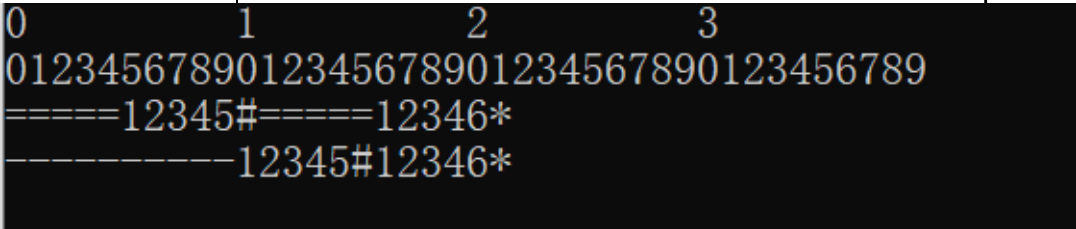
结论:

1、setw指定的宽度是总宽度，对于实型数据，\_包含\_\_\_\_(包含/不包含)小数点



# §. 基础知识题 – 输入输出部分

## 8、在cout中使用格式化控制符 K. setw+setfill的使用

<pre>#include &lt;iostream&gt; #include &lt;iomanip&gt;  using namespace std; int main() {     int a = 12345;      cout &lt;&lt; "0          1          2          3" &lt;&lt; endl;     cout &lt;&lt; "0123456789012345678901234567890123456789" &lt;&lt; endl;      cout &lt;&lt; setfill('=') &lt;&lt; setw(10) &lt;&lt; a &lt;&lt; '#' &lt;&lt; setw(10) &lt;&lt; a + 1 &lt;&lt; '*' &lt;&lt; endl;     cout &lt;&lt; setw(15) &lt;&lt; setfill('-') &lt;&lt; a &lt;&lt; '#' &lt;&lt; a + 1 &lt;&lt; '*' &lt;&lt; endl;      return 0; }</pre>	<p>运行截图:</p> 
--	--

结论:

- 1、setfill的作用是\_\_\_\_\_设置填充字符，将多余的空格位填充为某个字符\_\_\_\_\_
- 2、setfill的设置后，对后面的\_\_所有\_\_\_\_\_ (仅一个/所有)数据有效
- 3、解释为什么第4行的第个数(12346)前面没有-  
setw只对其后面的一个数据生效，第4行的a+1并没有设置宽度，所以没有空格给setfill填充-。





# § . 基础知识题 – 输入输出部分

## 8、在cout中使用格式化控制符

L.setw/setfill与ios::left/ios::right的混合使用 – (1)

<pre>#include &lt;iostream&gt; #include &lt;iomanip&gt;  using namespace std; int main() {     int a = 12345;     cout &lt;&lt; "0      1      2      3" &lt;&lt; endl;     cout &lt;&lt; "0123456789012345678901234567890123456789" &lt;&lt; endl;     cout &lt;&lt; setw(10) &lt;&lt; a &lt;&lt; '#' &lt;&lt; setw(10) &lt;&lt; a + 1 &lt;&lt; '*' &lt;&lt; endl;     cout &lt;&lt; setiosflags(ios::left);     cout &lt;&lt; setw(10) &lt;&lt; a &lt;&lt; '#' &lt;&lt; setw(10) &lt;&lt; a + 1 &lt;&lt; '*' &lt;&lt; endl;     return 0; }</pre>		运行截图:
<pre>#include &lt;iostream&gt; #include &lt;iomanip&gt;  using namespace std; int main() {     int a = 12345;     cout &lt;&lt; "0      1      2      3" &lt;&lt; endl;     cout &lt;&lt; "0123456789012345678901234567890123456789" &lt;&lt; endl;     cout &lt;&lt; setfill('=') &lt;&lt; setw(10) &lt;&lt; a &lt;&lt; '#' &lt;&lt; setw(10) &lt;&lt; a + 1 &lt;&lt; '*' &lt;&lt; endl;     cout &lt;&lt; setiosflags(ios::left);     cout &lt;&lt; setfill('=') &lt;&lt; setw(10) &lt;&lt; a &lt;&lt; '#' &lt;&lt; setw(10) &lt;&lt; a + 1 &lt;&lt; '*' &lt;&lt; endl;     return 0; }</pre>		运行截图:





# § . 基础知识题 – 输入输出部分

## 8、在cout中使用格式化控制符

L.setw/setfill与ios::left/ios::right的混合使用 – (2) – 同时使用(错误)

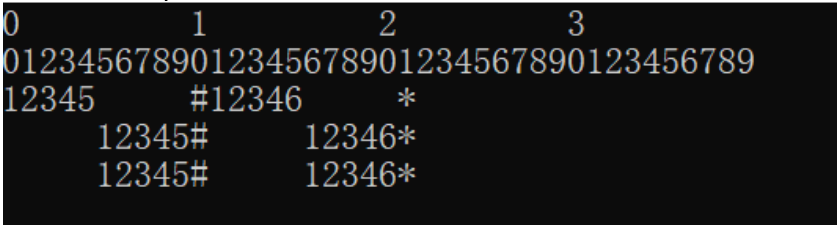
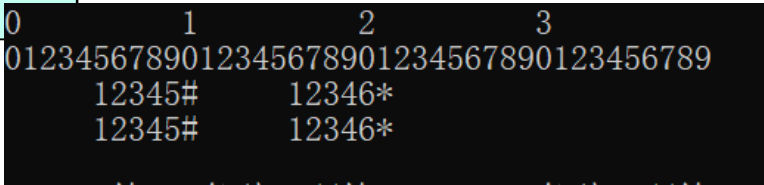
<pre>#include &lt;iostream&gt; #include &lt;iomanip&gt; using namespace std; int main() {     int a = 12345;     cout &lt;&lt; "0          1          2          3" &lt;&lt; endl;     cout &lt;&lt; "0123456789012345678901234567890123456789" &lt;&lt; endl;     /* 左对齐 */     cout &lt;&lt; setiosflags(ios::left) &lt;&lt; setw(10) &lt;&lt; a &lt;&lt; '#' &lt;&lt; setw(10) &lt;&lt; a + 1 &lt;&lt; '*' &lt;&lt; endl;     /* 右对齐 */     cout &lt;&lt; setiosflags(ios::right) &lt;&lt; setw(10) &lt;&lt; a &lt;&lt; '#' &lt;&lt; setw(10) &lt;&lt; a + 1 &lt;&lt; '*' &lt;&lt; endl;     /* 左对齐 */     cout &lt;&lt; setiosflags(ios::left) &lt;&lt; setw(10) &lt;&lt; a &lt;&lt; '#' &lt;&lt; setw(10) &lt;&lt; a + 1 &lt;&lt; '*' &lt;&lt; endl;     return 0; }</pre>	<p>运行截图:</p>
<pre>#include &lt;iostream&gt; #include &lt;iomanip&gt; using namespace std; int main() {     int a = 12345;     cout &lt;&lt; "0          1          2          3" &lt;&lt; endl;     cout &lt;&lt; "0123456789012345678901234567890123456789" &lt;&lt; endl;     /* 右对齐 */     cout &lt;&lt; setiosflags(ios::right) &lt;&lt; setw(10) &lt;&lt; a &lt;&lt; '#' &lt;&lt; setw(10) &lt;&lt; a + 1 &lt;&lt; '*' &lt;&lt; endl;     /* 左对齐 */     cout &lt;&lt; setiosflags(ios::left) &lt;&lt; setw(10) &lt;&lt; a &lt;&lt; '#' &lt;&lt; setw(10) &lt;&lt; a + 1 &lt;&lt; '*' &lt;&lt; endl;     return 0; }</pre>	<p>运行截图:</p>



# § . 基础知识题 - 输入输出部分

## 8、在cout中使用格式化控制符

L. setw/setfill与ios::left/ios::right的混合使用 - 在上一页的基础上将程序改正确，并给出截图

<pre>#include &lt;iostream&gt; #include &lt;iomanip&gt; using namespace std; int main() {     int a = 12345;     cout &lt;&lt; "0      1      2      3" &lt;&lt; endl;     cout &lt;&lt; "0123456789012345678901234567890123456789" &lt;&lt; endl;     /* 左对齐 */     cout &lt;&lt; setiosflags(ios::left) &lt;&lt; setw(10) &lt;&lt; a &lt;&lt; '#' &lt;&lt; setw(10) &lt;&lt; a + 1 &lt;&lt; '*' &lt;&lt; endl;     /* 右对齐 */     cout &lt;&lt; setiosflags(ios::right) &lt;&lt; setw(10) &lt;&lt; a &lt;&lt; '#' &lt;&lt; setw(10) &lt;&lt; a + 1 &lt;&lt; '*' &lt;&lt; endl;     /* 左对齐 */     cout &lt;&lt; setiosflags(ios::left) &lt;&lt; setw(10) &lt;&lt; a &lt;&lt; '#' &lt;&lt; setw(10) &lt;&lt; a + 1 &lt;&lt; '*' &lt;&lt; endl;     return 0; }</pre>	<p>运行截图:</p> 
<pre>#include &lt;iostream&gt; #include &lt;iomanip&gt; using namespace std; int main() {     int a = 12345;     cout &lt;&lt; "0      1      2      3" &lt;&lt; endl;     cout &lt;&lt; "0123456789012345678901234567890123456789" &lt;&lt; endl;     /* 右对齐 */     cout &lt;&lt; setiosflags(ios::right) &lt;&lt; setw(10) &lt;&lt; a &lt;&lt; '#' &lt;&lt; setw(10) &lt;&lt; a + 1 &lt;&lt; '*' &lt;&lt; endl;     /* 左对齐 */     cout &lt;&lt; setiosflags(ios::left) &lt;&lt; setw(10) &lt;&lt; a &lt;&lt; '#' &lt;&lt; setw(10) &lt;&lt; a + 1 &lt;&lt; '*' &lt;&lt; endl;     return 0; }</pre>	<p>结论: 如果想要right对齐后再left对齐, 需要在两者之间加入一句: _____cout&lt;&lt;resetiosflags(ios::right);_____</p> <p>运行截图:</p> 



## §. 基础知识题 - 输入输出部分

此页不要删除，也没有意义，仅仅为了分隔题目



# §. 基础知识题 – 输入输出部分

## 9、在cin中使用格式化控制符

A. 基本要求：修改cin语句，使能从键盘输入16进制数，并仿照第2题，构造若干测试组数据（多个Page），回答相应问题

```
#include <iostream>
#include <iomanip>

using namespace std;
int main()
{
    short a;    //程序允许修改此句
    cin >> a;    //程序允许修改此句

    cout << "dec:" << dec << a << endl;
    cout << "hex:" << hex << a << endl;
    cout << "oct:" << oct << a << endl;

    return 0;
}
```

- 名词约定：
- 输入正确 - 指数学上合法的数，但不代表一定在C/C++的某类型数据的数据范围内（下同）
  - 非法字符 - 指对short型的16进制数而言是非法的字符
  - 输入正数 - 指键盘输入的16进制数，以十进制输出时，变量的值显示为“xx”形式
  - 输入负数 - 指键盘输入的16进制数，以十进制输出时，变量的值显示为“-xx”形式
9. A. a - 变量为short型，前面是输入正确且范围合理的正数，后面加回车、空格、非法字符
9. A. b - 变量为short型，验证16进制方式能否以“-xx”的方式输入负数
9. A. c - 变量为short型，16进制方式输入负数的正确方法
9. A. d - 变量为short型，输入正确，但超short型范围
9. A. e - 变量为u\_short型，输入正确，但超u\_short型范围
9. A. f - 你自己学习过程中发现的、你认为需要写上的测试及总结（本项不强求）
- 出于减轻工作量的考虑，下面各项不要求：
- ★ 不需要加int/u\_int的测试数据
  - ★ 不再需要和同类型赋值去比较
  - ★ 不考虑cin连续读入多个16进制数时，中间含非法字符的问题（整个第3题）
  - ★ 上述项虽然不强制在作业中体现出来，但是自己一定要做一遍，跟第2/3题的对应项的结论去对比
  - ★ 本次作业完成后，默认你已掌握这些作业不要求的知识点!!!



# Aa

```
short a;  
cin >> a;  
cout << "dec:" << dec << a << endl;  
cout << "hex:" << hex << a << endl;  
cout << "oct:" << oct << a << endl;  
return 0;
```

Microsoft Visual Studio 调试控制台

```
28  
dec:28  
hex:1c  
oct:34
```

```
28 m  
dec:28  
hex:1c  
oct:34
```



# Ab

```
short a;  
cin >>hex>>a;  
cout << "dec:" << dec << a << endl;  
cout << "hex:" << hex << a << endl;  
cout << "oct:" << oct << a << endl;  
return 0;
```

Microsoft Visu

```
-4c  
dec:-76  
hex:ffb4  
oct:177664
```

F:\C++练习\各种  
要在调试停止时  
按任意键关闭此

16进制可以通过输入-xx来得到负数的十进制



# Ac

说明一点，举个例子，想输出-1，我现在是不能在键盘上输入ffffffff而得到-1的，输入ffffffff只会显示超范围，所以说这个cin是不能把二进制补码第一位当成符号位的，那么如果我想输入16进制数而得到-1，我输入的应该也是-1，但是输出时我可以规定16进制形式输出，会发现得到的是ffffffff。

```
-1  
dec:-1  
hex:ffffffff  
oct:3777777777
```

所以总结的话还是说，不能通过输入补码的16进制的符号位为-1（也就是ffffffff）去得到-1，但是可以通过直接输入-1来得到-1.

```
ffffffff  
dec:2147483647  
hex:7fffffffff  
oct:1777777777
```



## Ad

```
short a;  
cin >> a;  
cout << "dec:" << dec << a << endl;  
cout << "hex:" << hex << a << endl;  
cout << "oct:" << oct << a << endl;  
return 0;
```

Microsoft  
40000  
dec:32767  
hex:7fff  
oct:77777  
F:\C++练习\  
要在调试停  
按任意键关

## Ae

```
unsigned short a;  
cin >> a;  
cout << "dec:" << dec << a << endl;  
cout << "hex:" << hex << a << endl;  
cout << "oct:" << oct << a << endl;  
return 0;
```

70000  
dec:65535  
hex:ffff  
oct:177777  
F:\C++练习\  
要在调试停  
按任意键关





# Af

`cin`的输入与直接赋值是有区别的，`cin`的输入不会考虑二进制补码转化问题，不会因为你输入超过了某固定变量的数据范围而去转化成范围里的数，所以这也就解释了为什么不能输入ffffffff而得到-1，但是直接赋值就有很大的不同，赋值是可以将变量赋值为ffffffff而得到10进制表达-1的。



# §. 基础知识题 – 输入输出部分

## 9、在cin中使用格式化控制符

B. 基本要求：修改cin语句，使能从键盘输入8进制数，并仿照第2题，构造若干测试组数据（多个Page），回答相应问题

```
#include <iostream>
#include <iomanip>

using namespace std;
int main()
{
    int a;      //程序允许修改此句
    cin >> a;   //程序允许修改此句

    cout << "dec:" << dec << a << endl;
    cout << "hex:" << hex << a << endl;
    cout << "oct:" << oct << a << endl;

    return 0;
}
```

- 名词约定：
- 输入正确 - 指数学上合法的数，但不代表一定在C/C++的某类型数据的数据范围内（下同）
  - 非法字符 - 指对int型的8进制数而言是非法的字符
  - 输入正数 - 指键盘输入的8进制数，以十进制输出时，变量的值显示为“xx”形式
  - 输入负数 - 指键盘输入的8进制数，以十进制输出时，变量的值显示为“-xx”形式
9. B. a - 变量为int型，前面是输入正确且范围合理的正数，后面加回车、空格、非法字符
9. B. b - 变量为int型，验证16进制方式能否以“-xx”的方式输入负数
9. B. c - 变量为int型，8进制方式输入负数的正确方法
9. B. d - 变量为int型，输入正确，但超int型范围
9. B. e - 变量为u\_int型，输入正确，但超u\_int型范围
9. B. f - 你自己学习过程中发现的、你认为需要写上的测试及总结（本项不强求）
- 出于减轻工作量的考虑，下面各项不要求：
- ★ 不需要加int/u\_int的测试数据
  - ★ 不再需要和同类型赋值去比较
  - ★ 不考虑cin连续读入多个8进制数时，中间含非法字符的问题（整个第3题）
  - ★ 上述项虽然不强制在作业中体现出来，但是自己一定要做一遍，跟第2/3题的对应项的结论去对比
  - ★ 本次作业完成后，默认你已掌握这些作业不要求的知识点!!!



# Ba

```
int a;  
cin >> a;  
cout << "dec:" << dec << a << endl;  
cout << "hex:" << hex << a << endl;  
cout << "oct:" << oct << a << endl;  
  
return 0;
```

A screenshot of a Windows command prompt window titled "Micro:". The window shows the output of the C++ program: 45, dec:45, hex:2d, and oct:55. Below the output, there is a prompt "F:\C++练" and a message "要在调试" and "按任意键".

45  
dec:45  
hex:2d  
oct:55  
F:\C++练  
要在调试  
按任意键

A large, stylized representation of the program's output, showing the values 45, dec:45, hex:2d, and oct:55 in a large, colorful font on a black background.

45 a  
dec:45  
hex:2d  
oct:55



## Bb

```
int a;  
cin >> oct>>a;  
cout << "dec:" << dec << a << endl;  
cout << "hex:" << hex << a << endl;  
cout << "oct:" << oct << a << endl;
```

```
return 0;
```

Microsoft Visual Studio 调试控制台

```
-56  
dec:-46  
hex:ffffffd2  
oct:3777777722
```

8进制可以通过输入-xx来得到负数的十进制



# Bc

这个就基本与前面的Ac类似，还是刚才那个问题，不能通过符号位去确定负数，但是可以在输入时就加负号来输出负数。



## Bd

```
int a;  
cin >> a;  
cout << "dec:" << dec << a << endl;  
cout << "hex:" << hex << a << endl;  
cout << "oct:" << oct << a << endl;
```

```
return 0;
```

Microsoft Visual Studio 调试控制台

```
3000000000  
dec:2147483647  
hex:7fffffff  
oct:17777777777
```

## Be

```
unsigned int a;
```

```
cin >> a;  
cout << "dec:" << dec << a << endl;  
cout << "hex:" << hex << a << endl;  
cout << "oct:" << oct << a << endl;
```

```
return 0;
```

Microsoft Visual Studio 调试控制台

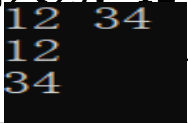
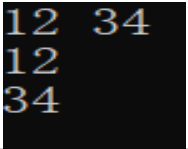
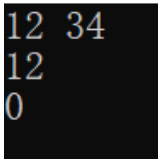
```
5000000000  
dec:4294967295  
hex:ffffffff  
oct:37777777777
```



# § . 基础知识题 – 输入输出部分

## 9、在cin中使用格式化控制符

### C. 格式控制符setiosflags(ios::skipws)的使用

<pre>#include &lt;iostream&gt; using namespace std;  int main() {     int a, b;      cin &gt;&gt; a &gt;&gt; b;      cout &lt;&lt; a &lt;&lt; endl;     cout &lt;&lt; b &lt;&lt; endl;     return 0; }</pre>	<pre>#include &lt;iostream&gt; #include &lt;iomanip&gt; using namespace std; int main() {     int a, b;     cin &gt;&gt; setiosflags(ios::skipws);     cin &gt;&gt; a &gt;&gt; b;     cout &lt;&lt; a &lt;&lt; endl;     cout &lt;&lt; b &lt;&lt; endl;     return 0; }</pre>	<pre>#include &lt;iostream&gt; #include &lt;iomanip&gt; using namespace std; int main() {     int a, b;     cin.unsetf(ios::skipws);     cin &gt;&gt; a &gt;&gt; b;     cout &lt;&lt; a &lt;&lt; endl;     cout &lt;&lt; b &lt;&lt; endl;     return 0; }</pre>
假设键盘输入为: 12 34✓ 则输出为: 	假设键盘输入为: 12 34✓ 则输出为: 	假设键盘输入为: 12 34✓ 则输出为: 
<p>1、“不忽略前导空格”的意思，是空格不作为_分隔符_，而是做为_空格符，在这里相当于给一个整型变量赋值空格符_（因此导致第3个例子b未取得34）</p> <p>2、setiosflags(ios::skipws)在缺省情况下是_有效___(有效/无效)的，即不设置也生效</p> <p>3、如果想取消“忽略前导空格”的设置，应使用__resetiosflags(ios::skipws)_____</p>		



## §. 基础知识题 - 输入输出部分

此页不要删除，也没有意义，仅仅为了分隔题目





## §. 基础知识题 – 输入输出部分

### 10、C语言的scanf/printf的使用

- 1、阅读附件给出的补充资料，自行尝试上面的示例（部分不清楚的内容可以暂时放一放）
- 2、用C语言的printf函数完成与C++的cout一样的输出（不需要输出中文提示部分）  
说明：① 附件中的3-bl.cpp已包含了完整的cout输出，不准改动只准改动首行的个人信息以及printf函数所在行，使其与上面cout的输出要求完全一致即可  
② 按回车键依次执行，直到全部结束  
③ C语言补充资料中无printf对整数带符号位(+)的输出方法，需自行查找相关资料  
(不允许用if-else之类的分支语句或条件表达式(?:)输出符号位，否则得分直接为0)



## §. 基础知识题 - 输入输出部分

此页不要删除，也没有意义，仅仅为了分隔题目