

§ . 基础知识题 - C++方式输入输出的格式化控制



要求:

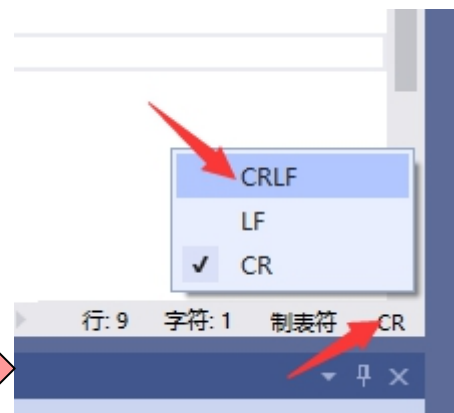
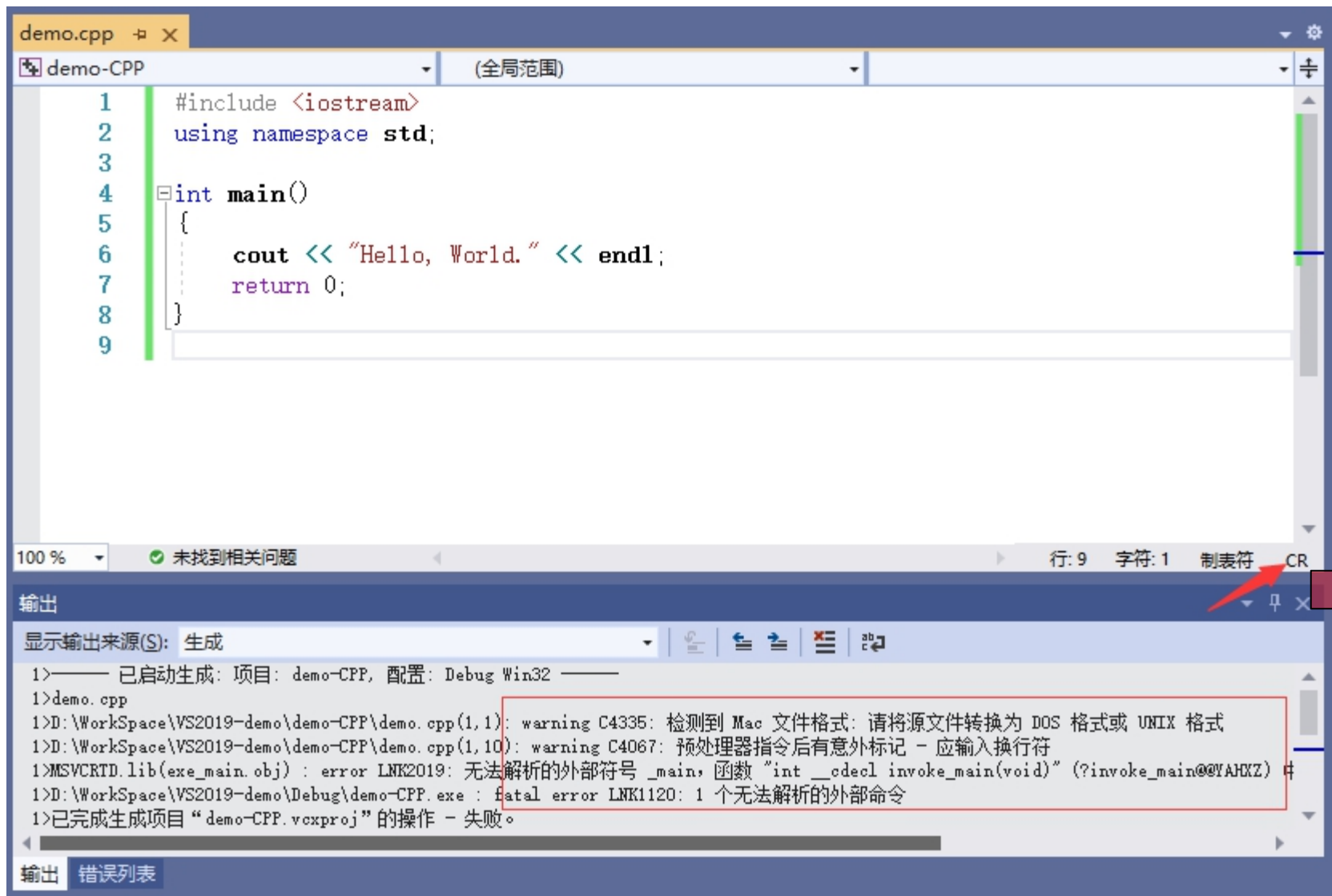
- 1、完成本文档中所有的题目并写出分析、运行结果
- 2、无特殊说明，均使用VS2019编译即可
- 3、直接在本文件上作答，**写出答案/截图（不允许手写、手写拍照截图）**即可；填写答案时，为适应所填内容或贴图，**允许调整**页面的字体大小、颜色、文本框的位置等
 - ★ 在保证一页一题的前提下，具体页面布局可以自行发挥，简单易读即可
 - ★ **不允许**手写在纸上，再拍照贴图
 - ★ **允许**在各种软件工具上手写完成，再截图贴图
- 4、转换为pdf后提交
- 5、**3月24日前**网上提交本次作业（在“实验报告”中提交）

§. 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VS2019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

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 - C++方式输入输出的格式化控制

1、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()
{
```

```
    short a1 = 1234, a2 = 0x1234, a3 = 01234, a4 = 0b1101001; //常量为各进制表示正数
```

```
    cout << "dec:" << dec << a1 << ' ' << a2 << ' ' << a3 << ' ' << a4 << endl;
```

```
#include <iostream>
#include <iomanip>
using namespace std;
```

```
int main()
```

```
{
    short a1 = 1234, a2 = 0x1234, a3 = 01234, a4 = 0b1101001; //常量为各进制表示正数
    cout << "dec:" << dec << a1 << ' ' << a2 << ' ' << a3 << ' ' << a4 << endl;
    cout << "hex:" << hex << a1 << ' ' << a2 << ' ' << a3 << ' ' << a4 << endl;
    cout << "oct:" << oct << a1 << ' ' << a2 << ' ' << a3 << ' ' << a4 << endl;
    cout << endl;
```

```
    short b1 = -1234, b2 = -0x1234, b3 = -01234, b4 = -0b1101001; //常量为各进制表示负数
```

```
    cout << "dec:" << dec << b1 << ' ' << b2 << ' ' << b3 << ' ' << b4 << endl;
```

```
    cout << "hex:" << hex << b1 << ' ' << b2 << ' ' << b3 << ' ' << b4 << endl;
```

```
    cout << "oct:" << oct << b1 << ' ' << b2 << ' ' << b3 << ' ' << b4 << endl;
```

```
    cout << endl;
```

```
    short c1 = 40000, c2 = 0x9876, c3 = 0171234, c4 = 0b1101010100111100; //赋值后最高位均为1, 有warning
```

```
    cout << "dec:" << dec << c1 << ' ' << c2 << ' ' << c3 << ' ' << c4 << endl;
```

```
    cout << "hex:" << hex << c1 << ' ' << c2 << ' ' << c3 << ' ' << c4 << endl;
```

```
    cout << "oct:" << oct << c1 << ' ' << c2 << ' ' << c3 << ' ' << c4 << endl;
```

```
    cout << endl;
```

```
    return 0;
```

```
}
```

未找到相关问题

```
Microsoft Visual Studio 调试控制台
dec:1234 4660 668 105
hex:4d2 1234 29c 69
oct:2322 11064 1234 151

dec:-1234 -4660 -668 -105
hex:fb2e edcc fd64 ff97
oct:175456 166714 176544 177627

dec:-25536 -26506 -3428 -10948
hex:9c40 9876 f29c d53c
oct:116100 114166 171234 152474

C:\Users\Administrator.PC-20190814YQ0J\source\repos\高程03\Debug\高程03.exe (进程 12108) 已退出, 代码为 0。
按任意键关闭此窗口...
```

//允许贴图覆盖代码部分



§. 基础知识题 - C++方式输入输出的格式化控制

1、在cout中使用格式化控制符

A. 的总结及结论:

- 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__二进制__形式
- 3、如果想使数据输出时使用不同进制，要加__dec, hex, oct__等进制前导符
- 4、输出__无__(有/无)二进制前导符
- 5、只有__十__进制有负数形式输出；
16进制输出负数时，特征是__转换为二进制补码形式，再从补码按无符号形式输出__16进制数__；
8进制输出负数时，特征是__转换为二进制补码形式，再从补码按无符号形式输出__8进制数__。



§. 基础知识题 - C++方式输入输出的格式化控制

1、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;
}
```

The screenshot shows the Microsoft Visual Studio IDE. On the left, the C++ source code is displayed, which is identical to the code block on the left. On the right, the 'Microsoft Visual Studio 调试控制台' (Debug Console) is open, showing the output of the program. The output consists of five lines of text, each representing the output of a cout statement. The first line shows the decimal values of a, a+1, and a+2. The second line shows the hexadecimal values. The third line shows the octal values. The fourth line shows the decimal values again. The fifth line shows the decimal values again. The output is as follows:

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

Below the output, the console shows the message: 'C:\Users\Administrator.PC-20190814\Programs\10984)已退出, 代码为 0。按任意键关闭此窗口。...' (C:\Users\Administrator.PC-20190814\Programs\10984)已退出, 代码为 0。按任意键关闭此窗口。...)

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 - C++方式输入输出的格式化控制

1、在cout中使用格式化控制符

C. setbase的使用：同1. A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)

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

using namespace std;
int main()
{
    return 0;
}
```

自行构造若干组测试数据，运行并截图

结论：

1、setbase中允许的合法值有__8, 10, 16__

2、当setbase中出现非法值时，处理方法是__按十进制输出__

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

```
#include <iostream>
#include <iomanip>
using namespace std;
int main()
{
    int a = 666;
    cout << "一进制输出" << setbase(1) << a << endl;
    cout << "二进制输出" << setbase(2) << a << endl;
    cout << "三进制输出" << setbase(3) << a << endl;
    cout << "八进制输出" << setbase(8) << a << endl;
    cout << "十进制输出" << setbase(10) << a << endl;
    cout << "十六进制输出" << setbase(16) << a << endl;
    cout << setbase(8);
    cout << a << " " << a + 1 << " " << a + 2 << endl;
    cout << setbase(10);
    cout << a << " " << a + 1 << " " << a + 2 << endl;
    cout << setbase(16);
    cout << a << " " << a + 1 << " " << a + 2 << endl;
    return 0;
}
```

Microsoft Visual Studio 调试控制台

```
一进制输出666
二进制输出666
三进制输出666
八进制输出1232
十进制输出666
十六进制输出29a
1232 1233 1234
666 667 668
29a 29b 29c
C:\Users\Administrator.PC-2019
程 7336)已退出，代码为 0。
按任意键关闭此窗口...
```

//构造的程序要求能看出对右侧问题的回答
//允许将构造的程序直接贴图上来，允许多页



§. 基础知识题 - C++方式输入输出的格式化控制

1、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()
{
    return 0;
}
```

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

using namespace std;
int main()
{
    int a=9999;
    cout << a << " " << a+1 << " " << a+2 << endl;
    cout << setiosflags(ios::uppercase)<<hex;
    cout << a << " " << a+1 << " " << a+2 << endl;
    cout << setiosflags(ios::uppercase) << oct;
    cout << a << " " << a+1 << " " << a+2 << endl;
    cout << setiosflags(ios::uppercase) << dec;
    cout << a << " " << a+1 << " " << a+2 << endl;
    return 0;
}
```

```
Microsoft Visual Studio 调试控制台

9999 10000 10001
270F 2710 2711
23417 23420 23421
9999 10000 10001

C:\Users\Administrator.PC-201
程 1616)已退出，代码为 0。
按任意键关闭此窗口...
```

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_十六_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_cout << 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)

//构造的程序要求能看出对右侧问题的回答
//允许将构造的程序直接贴图上来



§. 基础知识题 - C++方式输入输出的格式化控制

1、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()
{
    return 0;
}
```

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

using namespace std;
int main()
{
    int a=10;
    cout << a << " " << a+1 << " " << a+2 << endl;
    cout << setiosflags(ios::showpos) << hex;
    cout << a << " " << a+1 << " " << a+2 << endl;
    cout << setiosflags(ios::showpos) << oct;
    cout << a << " " << a+1 << " " << a+2 << endl;
    cout << setiosflags(ios::showpos) << dec;
    cout << a << " " << a+1 << " " << a+2 << endl;
    return 0;
}
```

Microsoft Visual Studio 调试控制台

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

C:\Users\Administrator.PC-20
程 1400)已退出, 代码为 0。
按任意键关闭此窗口...
```

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__十__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_cout << 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)

//构造的程序要求能看出对右侧问题的回答
//允许将构造的程序直接贴图上来



§. 基础知识题 - C++方式输入输出的格式化控制

1、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;
    float 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;
    cout << setprecision(7) << f1 << ' ' << f2 << endl;

    /* 第4组: 大于float型有效数字 */
    cout << endl;
    cout << setprecision(8) << f1 << ' ' << f2 << endl;
    cout << setprecision(9) << f1 << ' ' << f2 << endl;
    cout << setprecision(10) << f1 << ' ' << f2 << endl;
    cout << setprecision(25) << f1 << ' ' << f2 << endl;

    return 0;
}
```

本例贴图

```
Microsoft Visual Studio 调试控制台
1234.57 8765.43
1e+03 9e+03
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
C:\Users\Administrator.PC-20190814YQ0J\source\repos\高程03\Debug\高程03.exe (进程 9024)已退出, 代码为 0。
按任意键关闭此窗口...
```



§. 基础知识题 - C++方式输入输出的格式化控制

1、在cout中使用格式化控制符

F.setprecision的使用 - 单独使用 - (2)

```
#include <iostream>
#include <iomanip>
using namespace std;
int main()
{
    float f1 = 1234567890123456789.0F;
    float f2 = 9876543210987654321.0F;

    /* 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 并且 小与等于float型有效数字 */
    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;
    cout << setprecision(5) << f1 << ' ' << f2 << endl;
    cout << setprecision(6) << f1 << ' ' << f2 << endl;
    cout << setprecision(7) << f1 << ' ' << f2 << endl;

    /* 第3组: 大于float型有效数字 */
    cout << endl;
    cout << setprecision(8) << f1 << ' ' << f2 << endl;
    cout << setprecision(9) << f1 << ' ' << f2 << endl;
    cout << setprecision(10) << f1 << ' ' << f2 << endl; //为什么f1比f2少一位?
    cout << setprecision(11) << f1 << ' ' << f2 << endl;
    cout << setprecision(25) << f1 << ' ' << f2 << endl;

    return 0;
}
```

本例贴图

猜测在表示不可信数字时，若最后一位随机表示为0，则不输出，故f1少一位

```
Microsoft Visual Studio 调试控制台
1.23457e+18 9.87654e+18
1e+18 1e+19

1e+18 1e+19
1.2e+18 9.9e+18
1.23e+18 9.88e+18
1.235e+18 9.877e+18
1.2346e+18 9.8765e+18
1.23457e+18 9.87654e+18
1.234568e+18 9.876544e+18

1.2345679e+18 9.8765435e+18
1.23456794e+18 9.87654352e+18
1.23456794e+18 9.876543516e+18
1.2345679396e+18 9.8765435164e+18
1234567939550609408 9876543516404875264

C:\Users\Administrator.PC-20190814YQ0J\source\repos\高程03\Debug\高程03.exe (进程 6452)已退出, 代码为 0。
按任意键关闭此窗口...
```



§. 基础知识题 - C++方式输入输出的格式化控制

1、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;
    float f2 = 0.87654321F;

    /* 第1组: 不设或非法 */
    cout << f1 << ' ' << f2 << endl;
    cout << setprecision(0) << f1 << ' ' << f2 << endl;

    /* 第2组: 小与等于float型有效数字 */
    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;
    cout << setprecision(5) << f1 << ' ' << f2 << endl;
    cout << setprecision(6) << f1 << ' ' << f2 << endl;
    cout << setprecision(7) << f1 << ' ' << f2 << endl;

    /* 第3组: 大于float型有效数字 */
    cout << endl;
    cout << setprecision(8) << f1 << ' ' << f2 << endl;
    cout << setprecision(9) << f1 << ' ' << f2 << endl;
    cout << setprecision(10) << f1 << ' ' << f2 << endl;
    cout << setprecision(25) << f1 << ' ' << f2 << endl;

    return 0;
}
```

本例贴图

```
Microsoft Visual Studio 调试控制台
0.123457 0.876543
0.1 0.9

0.1 0.9
0.12 0.88
0.123 0.877
0.1235 0.8765
0.12346 0.87654
0.123457 0.876543
0.1234568 0.8765432

0.12345678 0.87654322
0.123456784 0.876543224
0.1234567836 0.8765432239
0.1234567835927009582519531 0.876543223857879638671875

C:\Users\Administrator.PC-20190814YQ0J\source\repos\高程03\Debug\高程03.exe (进程 9240) 已退出, 代码为 0。
按任意键关闭此窗口...
```



§. 基础知识题 - C++方式输入输出的格式化控制

1、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）

对于浮点数来说，n代表的就是有效数字，即若小数点前整数位有数字（不为0）时，n代表有效数字；若小数点前整数位为0，则n代表小数位数。

同时，在截断位数小于等于float型有效数字时，若截断位数后还有数字，则会根据这个数字对前一位进行四舍五入。在截断位数大于float型有效数字时，系统会按指定位数输出，但有效位数之后的数字都是不可信的。猜测F（2）中f1比f2少一位的原因是在表示不可信数字时，若最后一位随机表示为0，则不输出，故f1少一位。

2、将1.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）

经检验，double型同样适用



§. 基础知识题 - C++方式输入输出的格式化控制

1、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;
    float 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(25) << f1 << ' ' << f2 << endl;

    return 0;
}
```

贴图:

```
Microsoft Visual Studio 调试控制台

1234.57 8765.43
1234.567749 8765.431641

1234.6 8765.4
1234.5677 8765.4316
1234.5677490 8765.4316406
1234.5677490234 8765.4316406250
1234.56774902343750000000000000 8765.43164062500000000000000000

C:\Users\Administrator.PC-20190814Y00J\source\repos\高程03\Debug\高程03.exe (进程 11324)已退出, 代码为 0。
按任意键关闭此窗口...
```



§. 基础知识题 - C++方式输入输出的格式化控制

1、在cout中使用格式化控制符

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

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

using namespace std;
int main()
{
    float f1 = 1234567890123456789.0F;
    float f2 = 9876543210987654321.0F;

    /* 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:

```
Microsoft Visual Studio 调试控制台
1. 23457e+18 9.87654e+18
1234567939550609408.000000 9876543516404875264.000000

1234567939550609408.0 9876543516404875264.0
1234567939550609408.0000 9876543516404875264.0000
1234567939550609408.000000 9876543516404875264.000000
1234567939550609408.000000000 9876543516404875264.000000000
1234567939550609408.000000000000000000 9876543516404875264.000000000000000000
0

C:\Users\Administrator.PC-20190814YQ0J\source\repos\高程03\Debug\高程03.exe (进
程 12012)已退出, 代码为 0。
按任意键关闭此窗口...
```




§. 基础知识题 - C++方式输入输出的格式化控制

1、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;
    float 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:

```
Microsoft Visual Studio 调试控制台
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

C:\Users\Administrator.PC-20190814Y00J\source\repos\高程03\Debug\高程03.exe (进程 9012) 已退出, 代码为 0。
按任意键关闭此窗口...
```



§. 基础知识题 - C++方式输入输出的格式化控制

1、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）

对于浮点数来说，在setiosflags(ios::fixed)不设precision时，表示的数整数部分位数与原数相同，而不论原数小数部分有多少位，setiosflags(ios::fixed)之后，小数部分都将输出至小数点后6位，缺少数字则补0。在设置precision之后，setprecision (n)内表示的位数为小数点后的位数，不足则补0。

同样在截断位数小于等于float型有效数字时，若截断位数后还有数字，则会根据这个数字对前一位进行四舍五入。在截断位数大于float型有效数字时，系统会按指定位数输出，但有效位数之后的数字都是不可信的。

2、将1.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）

经检验，double型同样适用



§. 基础知识题 - C++方式输入输出的格式化控制

1、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;
    float 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(25) << f1 << ' ' << f2 << endl;

    return 0;
}
```

贴图:

```
Microsoft Visual Studio 调试控制台
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
1.23456774902343750000000000e+03 8.765431640625000000000000e+03

C:\Users\Administrator.PC-20190814YQ0J\source\repos\高程03\Debug\高程03.exe (进程 9168)已退出, 代码为 0。
按任意键关闭此窗口...
```



§. 基础知识题 - C++方式输入输出的格式化控制

1、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;
    float 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;

    return 0;
}
```

贴图:

```
Microsoft Visual Studio 调试控制台
1. 23457e+18 9.87654e+18
1. 234568e+18 9.87654e+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. 2345679395060940800e+18 9.87654351640487526400e+18

C:\Users\Administrator.PC-20190814YQ0J\source\repos\高程03\Debug\高程03.exe (进程 12104) 已退出, 代码为 0。
按任意键关闭此窗口...
```



§. 基础知识题 - C++方式输入输出的格式化控制

1、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;
    float 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;
}
```

贴图:

```
Microsoft Visual Studio 调试控制台
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

C:\Users\Administrator.PC-20190814YQ0J\source\repos\高程03\Debug\高程03.exe (进程 4188)已退出, 代码为 0。
按任意键关闭此窗口...
```



§. 基础知识题 - C++方式输入输出的格式化控制

1、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）

对于浮点数来说，在setiosflags(ios::scientific)不设precision时，不论原数位数有多少，设置后整数位输出统一为一位，小数位输出统一为六位。

在设置precision之后，setprecision(n)内表示的位数为小数点后的位数，不足则补0。

同样在截断位数小于等于float型有效数字时，若截断位数后还有数字，则会根据这个数字对前一位进行四舍五入。在截断位数大于float型有效数字时，系统会按指定位数输出，但有效位数之后的数字都是不可信的。

2、将1.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）

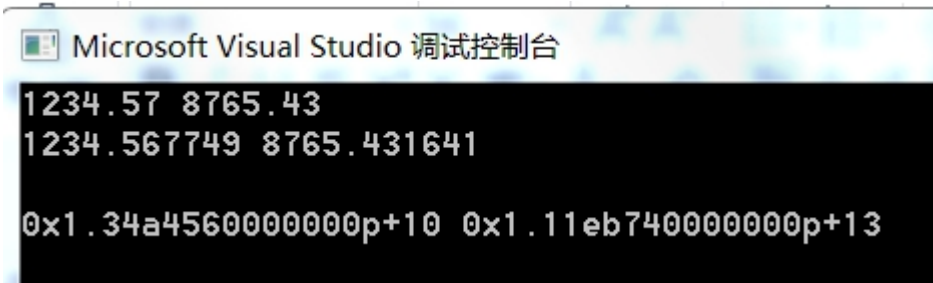

经检验，double型同样适用



§ . 基础知识题 - C++方式输入输出的格式化控制

1、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

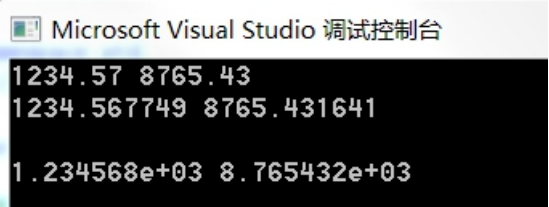
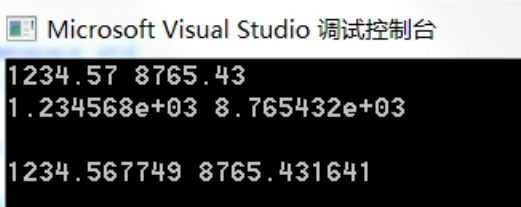
<pre>#include <iostream> #include <iomanip> using namespace std; int main() { float f1 = 1234.5678F, f2 = 8765.4321F; /* 第1组 */ cout << f1 << ' ' << f2 << endl; cout << setiosflags(ios::fixed) << f1 << ' ' << f2 << endl; /* 第2组 */ cout << endl; cout << setiosflags(ios::scientific) << f1 << ' ' << f2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



§ . 基础知识题 - C++方式输入输出的格式化控制

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



§ . 基础知识题 - C++方式输入输出的格式化控制

1、在cout中使用格式化控制符

J. setw的基本使用 - (1)

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

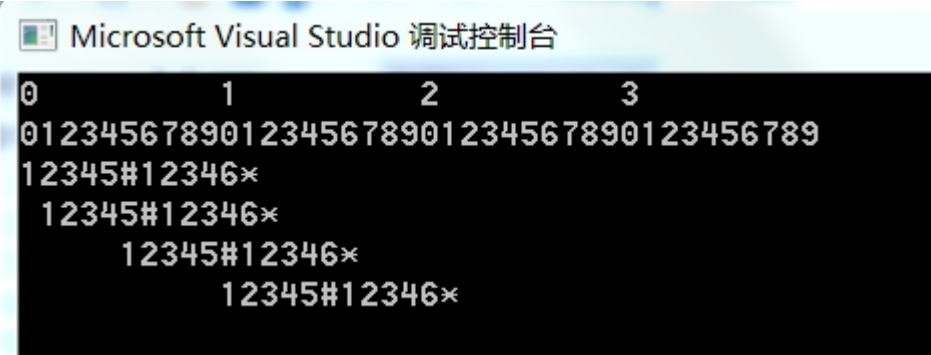
using namespace std;
int main()
{
    int a = 12345;

    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_输出函数后数据_；
当总宽度小于数据宽度时，显示规律为_差值部分在数据前用空格补齐，之后输出数据。_
- 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？ 为后几行输出的宽度提供基准，便于观察和计算后几行的宽度变化
- 4、每行输出的最后一个*，目的是什么？ 在每行输出的最后为宽度偏移提供基准，便于观察和比较宽度的变化



§. 基础知识题 - C++方式输入输出的格式化控制

1、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:



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 - C++方式输入输出的格式化控制

1、在cout中使用格式化控制符

K. setw+setfill的使用

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

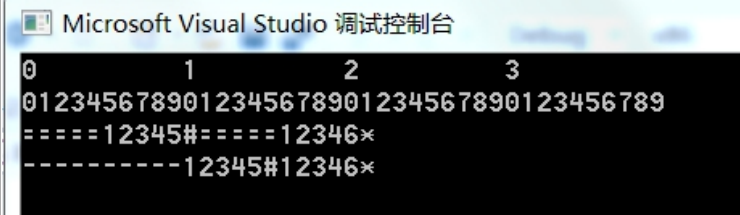
using namespace std;
int main()
{
    int a = 12345;

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

    cout << setfill('=') << setw(10) << a << '#' << setw(10) << a + 1 << '*' << endl;
    cout << setw(15) << setfill('-') << a << '#' << a + 1 << '*' << endl;

    return 0;
}
```

运行截图:



结论:

- 1、setfill的作用是__当setw总宽度小于数据宽度时，差值部分用setfill内所给符号补齐。_____
- 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2个数(12346)前面没有-
setw函数对后面的仅一个数据有效。第四行只有一个setw，已经作用于了第一个数，故对第二个数不起作用。



§ . 基础知识题 - C++方式输入输出的格式化控制

1、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)

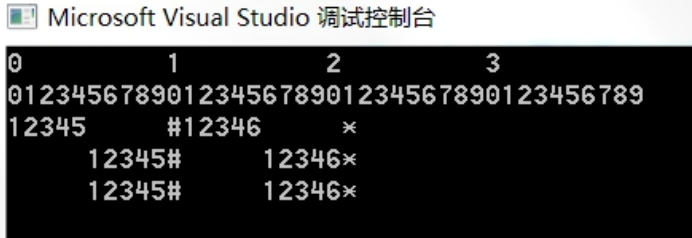
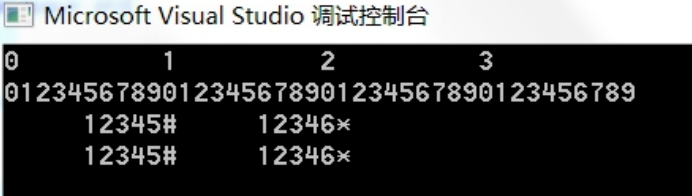
<pre>#include <iostream> #include <iomanip> using namespace std; int main() { int a = 12345; cout << "0 1 2 3" << endl; cout << "0123456789012345678901234567890123456789" << endl; cout << setw(10) << a << '#' << setw(10) << a + 1 << '*' << endl; cout << setiosflags(ios::left); cout << setw(10) << a << '#' << setw(10) << a + 1 << '*' << endl; return 0; }</pre>	运行截图:	
<pre>#include <iostream> #include <iomanip> using namespace std; int main() { int a = 12345; cout << "0 1 2 3" << endl; cout << "0123456789012345678901234567890123456789" << endl; cout << setfill('=') << setw(10) << a << '#' << setw(10) << a + 1 << '*' << endl; cout << setiosflags(ios::left); cout << setfill('=') << setw(10) << a << '#' << setw(10) << a + 1 << '*' << endl; return 0; }</pre>	<div>结论:</div> <div>1、ios::left的作用是__使输出左对齐__</div> <div>2、如果不设置, 缺省是_右对齐_(左/右对齐)</div>	运行截图:



§ . 基础知识题 - C++方式输入输出的格式化控制

1、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)

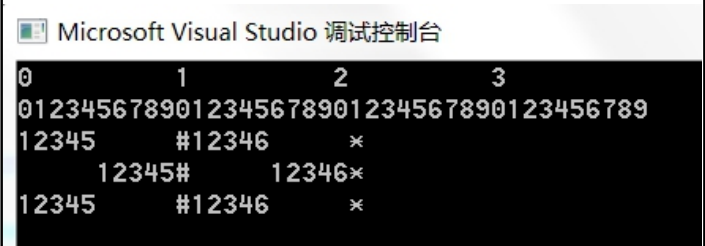
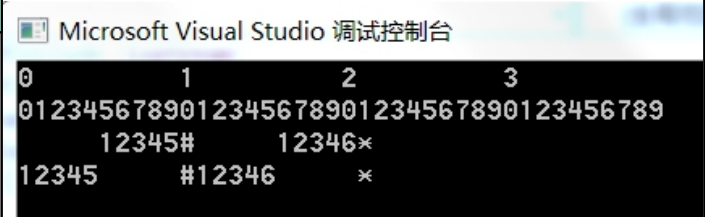
<pre>#include <iostream> #include <iomanip> using namespace std; int main() { int a = 12345; cout << "0 1 2 3" << endl; cout << "0123456789012345678901234567890123456789" << endl; /* 左对齐 */ cout << setiosflags(ios::left) << setw(10) << a << '#' << setw(10) << a + 1 << '*' << endl; /* 右对齐 */ cout << setiosflags(ios::right)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



§ . 基础知识题 - C++方式输入输出的格式化控制

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



§. 基础知识题 - C++方式输入输出的格式化控制

此页不要删除，也没有意义，仅仅为了分隔题目



§. 基础知识题 - C++方式输入输出的格式化控制

2、在cin中使用格式化控制符

A. 基本要求：从键盘输入16进制数

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

using namespace std;
int main()
{
    short a;
    cin >> hex >> a;

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

    return 0;
}
```

- 1、贴图即可，不需要写分析结果
- 2、暂不考虑输入错误

1、输入：1a2b✓（合理正数）

```
1a2b
dec:6699
hex:1a2b
oct:15053
```

2、输入：a1b2✓（超上限但未超同类型的unsigned上限）

```
a1b2
dec:32767
hex:7fff
oct:77777
```

3、输入：ffffff✓（超上限且超过同类型的unsigned上限）

```
ffffff
dec:32767
hex:7fff
oct:77777
```

4、输入：-1a2b✓（合理负数）

```
-1a2b
dec:-6699
hex:e5d5
oct:162725
```

5、输入：-ffffff✓（超下限）

```
-ffffff
dec:-32768
hex:8000
oct:100000
```



§. 基础知识题 - C++方式输入输出的格式化控制

2、在cin中使用格式化控制符

B. 基本要求：从键盘输入8进制数（自行构造测试数据）

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

using namespace std;
int main()
{
    int a;
    cin >> setbase(8) >> a;

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

    return 0;
}
```

- 1、贴图即可，不需要写分析结果
- 2、暂不考虑输入错误

1、输入：_666_ ✓ （合理正数）

```
666
dec:438
hex:1b6
oct:666
```

2、输入：20000000000 ✓ （超上限但未超同类的unsigned上限）

```
20000000000
dec:2147483647
hex:7fffffff
oct:17777777777
```

3、输入：20000000000 ✓ （超上限且超过同类的unsigned上限）

```
40000000000
dec:2147483647
hex:7fffffff
oct:17777777777
```

4、输入：_-1234_ ✓ （合理负数）

```
-1234
dec:-668
hex:ffffd64
oct:37777776544
```

5、输入：__-200000000004__ ✓ （超下限）

```
-200000000004
dec:-2147483648
hex:80000000
oct:20000000000
```



§ . 基础知识题 - C++方式输入输出的格式化控制

2、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 <iostream> using namespace std; int main() { int a, b; cin >> a >> b; cout << a << endl; cout << b << endl; return 0; }</pre>	<pre>#include <iostream> #include <iomanip> using namespace std; int main() { int a, b; cin >> setiosflags(ios::skipws); cin >> a >> b; cout << a << endl; cout << b << endl; return 0; }</pre>	<pre>#include <iostream> #include <iomanip> using namespace std; int main() { int a, b; cin.unsetf(ios::skipws); cin >> a >> b; cout << a << endl; cout << b << endl; return 0; }</pre>
假设键盘输入为: 12 34✓ 则输出为: __12 34__	假设键盘输入为: 12 34✓ 则输出为: __12 34__	假设键盘输入为: 12 34✓ 则输出为: __12 0__

综合以上三个例子可以得到如下结论:

- 1、“忽略前导空格”的意思, 是空格不作为一个字符, 而是做为有效分隔的符号 (因此导致第3个例子b未取得34)
- 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
- 3、如果想取消“忽略前导空格”的设置, 应使用____unsetf(ios::skipws);_____



§. 基础知识题 - C++方式输入输出的格式化控制

此页不要删除，也没有意义，仅仅为了分隔题目