



§. 基础知识题 - 与cout有关的成员函数的基本使用

要求:

- 1、完成本文档中所有的测试程序并填写运行结果，从而体会这些cin的流成员函数的用法及区别
- 2、题目明确指定编译器外，缺省使用VS2022即可
 - ★ 如果要换成其他编译器，可能需要自行修改头文件适配
 - ★ 部分代码编译时有warning，不影响概念理解，可以忽略
- 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、**11月24日前**网上提交本次作业（在“文档作业”中提交）

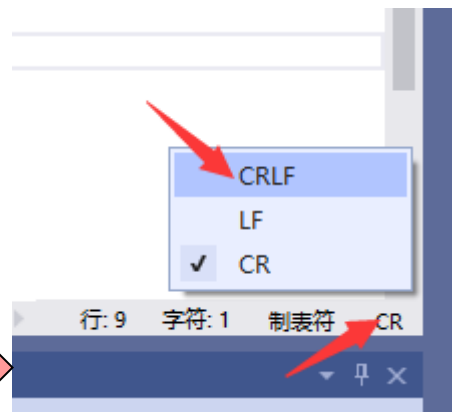
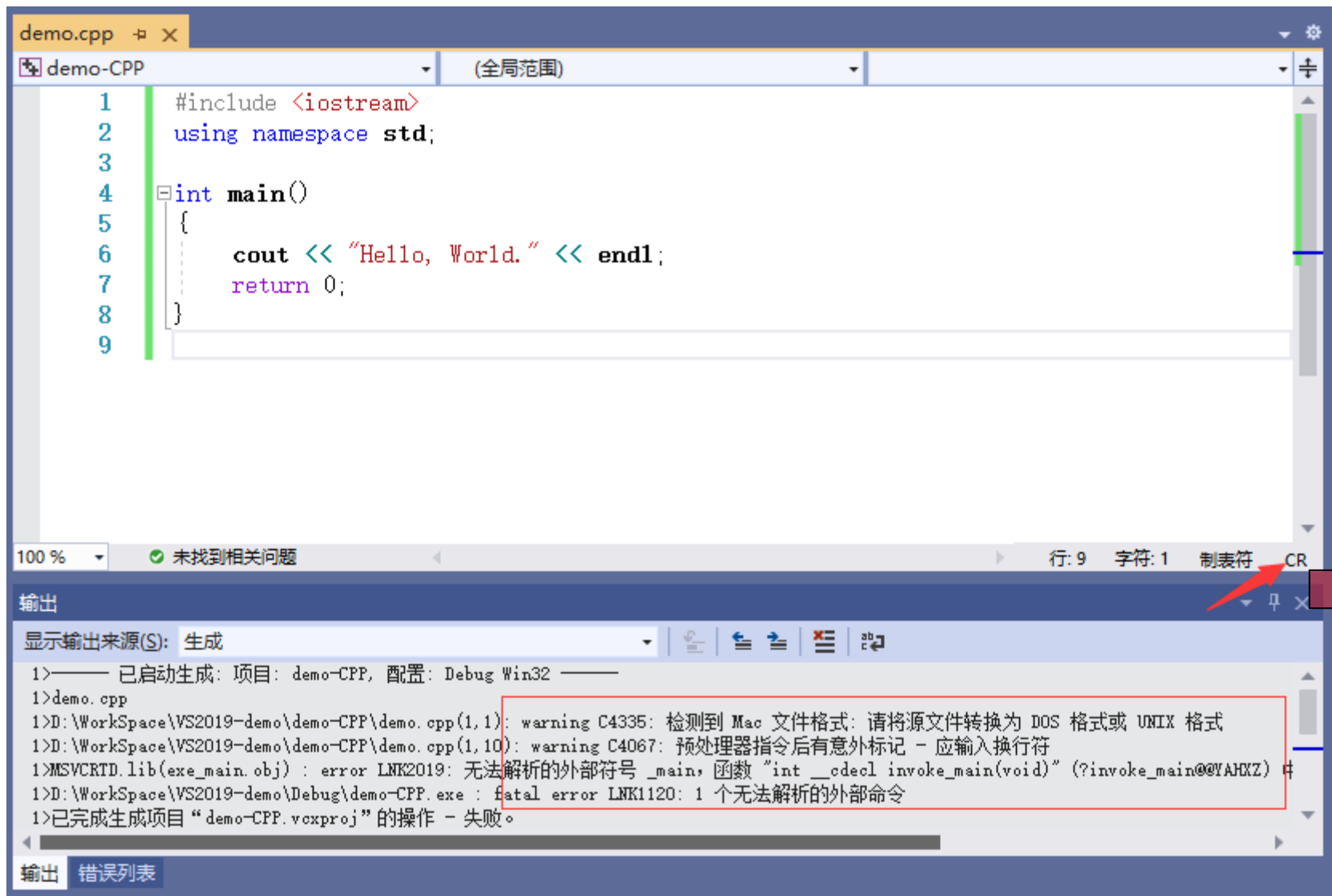


§. 基础知识题 - 与cout有关的成员函数的基本使用

注意:

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口的

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 - 与cout有关的成员函数的基本使用

1. 用于字符输出的流成员函数

★ `cout.put`(字符常量/字符变量)

功能：向标准输出设备输出一个字符

★ `cout.write`(字符串常量/变量, 输出长度)

功能：向标准输出设备输出n个字符（如果n超过串长，则输出串长）



§. 基础知识题 - 与cout有关的成员函数的基本使用

1. 用于字符输出的流成员函数

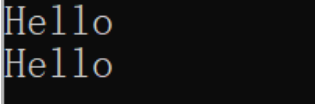
例1: cout.put()

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

int main()
{
    char str[] = "Hello";
    int i;

    for (i = 0; i < 5; i++)
        cout.put(str[i]);
    cout.put('\n');
    cout.put('H').put('e').put('l').put('l').put('o').put(0x0A);

    return 0;
}
```

运行结果: 



§. 基础知识题 - 与cout有关的成员函数的基本使用

1. 用于字符输出的流成员函数

例2: cout.write()

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

int main()
{
    char s1[] = "Hello";
    cout.write(s1, 5);
    cout.put('\n');
    cout.write(s1, 10);
    cout.put('\n');

    char s2[] = { 'H', 'e', 'l', 'l', 'o' };
    cout.write(s2, 5);
    cout.put('\n');
    cout.write(s2, 10);
    cout.put('\n');

    return 0;
}
```

运行结果.

```
Hello
Hello烫烫
Hello
Hello烫烫
```

当write的参数是字符串(s1)，且write要写的长度超过字符串长度时的表现：
输出完字符串后不会在输出到'\0'时停止输出，而是继续输出越界内容。

当write的参数非字符串(s2)，且write要写的长度超过字符串长度时的表现：
在输出完s2中的内容后，继续输出越界内容。

结论：用write向标准输出设备输出指定个数的字符时，输出缓冲区__不要求__(要求/不要求)是字符串



§ . 基础知识题 – 与cout有关的成员函数的基本使用

2. 用于字符输出控制的流成员函数

- ★ cout.setf(控制标记)
功能：设置指定的控制标记 (右表为常用)
- ★ cout.unsetf(控制标记)
功能：清除指定的控制标记 (右表为常用)
- ★ cout.width(宽度)
功能：设置指定的输出宽度
- ★ cout.fill(字符常量/字符变量)
功能：设置填充字节
- ★ cout.precision(精度)
功能：设置浮点数的输出精度

控制标记	作用
ios::fixed	设置浮点数以固定的小数位数显示
ios::scientific	设置浮点数以科学计数法（即指数形式）显示
ios::left	输出数据左对齐
ios::right	输出数据右对齐
ios::skipws	忽略前导的空格 (适用于cin, 不适用于cout)
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
ios::showpos	输出正数时，给出“+”号



§. 基础知识题 - 与cout有关的成员函数的基本使用

2. 用于字符输出控制的流成员函数

例3: cout.width()

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

int main()
{
    char s1[] = "Hello";

    cout << "01234567890123456789" << endl;
    cout.width(10);
    cout << s1 << '*' << endl;
    cout << s1 << '#' << endl;

    return 0;
}
```

运行结果:

```
01234567890123456789
      Hello*
Hello#
```

结论:

1、cout.width(10) 等价于 cout << setw(10)

2、cout.width() 设置后仅1次 (仅1次/始终) 有效



§. 基础知识题 - 与cout有关的成员函数的基本使用

2. 用于字符输出控制的流成员函数

例4: cout.width()与cout.fill()

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

int main()
{
    char s1[] = "Hello";

    cout << "01234567890123456789" << endl;
    cout.width(10);
    cout.fill('$');
    cout << s1 << '*' << endl;
    cout.width(15);
    cout << s1 << '#' << endl;
    cout.width(12);
    cout.fill(' ');
    cout << s1 << '*' << endl;

    return 0;
}
```

运行结果:

```
01234567890123456789
$$$$$Hello*
$$$$$$$$$$$Hello#
      Hello*
```

结论:

- 1、cout.fill()等价于 cout << `__setfill()`__
- 2、cout.fill()设置后_始终_(仅1次/始终)有效
- 3、默认的cout.fill()设置是哪个字符? ' ' (空格)



§. 基础知识题 - 与cout有关的成员函数的基本使用

2. 用于字符输出控制的流成员函数

例5: cout.width()与cout.setf()

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

int main()
{
    char s1[] = "Hello";

    cout << "01234567890123456789" << endl;
    cout.width(10);
    cout.setf(ios::left);
    cout << s1 << '*' << endl;
    cout.width(15);
    cout << s1 << '#' << endl;

    return 0;
}
```

运行结果:

```
01234567890123456789
Hello      *
Hello          #
```

结论:

1、cout.setf(ios::left) 等价于 cout << [setiosflags\(ios::left\)](#)_

2、cout.setf() 设置后__[始终](#)__(仅1次/始终)有效



§. 基础知识题 - 与cout有关的成员函数的基本使用

2. 用于字符输出控制的流成员函数

例6: cout.width()与cout.setf()

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

int main()
{
    char s1[] = "Hello";

    cout << "01234567890123456789" << endl;
    cout.width(10);
    cout.setf(ios::left);
    cout << s1 << '*' << endl;
    cout.setf(ios::right);
    cout.width(15);
    cout << s1 << '#' << endl;
    cout.width(10);
    cout.setf(ios::left);
    cout << s1 << '*' << endl;
    return 0;
}
```

运行结果:

```
01234567890123456789
Hello      *
           Hello#
Hello*
```

结论:

- 1、cout.setf(ios::left) 等价于 cout << `_setiosflags(ios::left)_`
- 2、cout.setf(ios::right) 等价于 cout << `_setiosflags(ios::left)_`
- 3、cout.setf() 设置后_始终_(仅1次/始终)有效
- 4、不设置默认是_右_(左/右)对齐
- 5、left后设置right是_有效_(有效/无效)的
- 6、right后设置left是_无效_(有效/无效)的



§. 基础知识题 - 与cout有关的成员函数的基本使用

2. 用于字符输出控制的流成员函数

例7: `cout.width()`与`cout.setf()`、`cout.unsetf()`

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

int main()
{
    char s1[] = "Hello";

    cout << "01234567890123456789" << endl;
    cout.width(10);
    cout.setf(ios::left);
    cout << s1 << '*' << endl;
    cout.setf(ios::right);
    cout.width(15);
    cout << s1 << '#' << endl;
    cout.width(10);
    cout.unsetf(ios::right);
    //此处添句话, 需用cout. 函数名
    cout.setf(ios::left);
    cout << s1 << '*' << endl;
    return 0;
}
```

将程序补充完整, 得到期望的运行结果:

```
01234567890123456789
Hello      *
           Hello#
Hello      *
```

所用的`cout._unsetf(ios::right);`等价于
`cout << _resetiosflags(ios::right);`

提示: 回忆并参考第3章的作业

本页需填写答案



§. 基础知识题 - 与cout有关的成员函数的基本使用

2. 用于字符输出控制的流成员函数

例8: cout.precision()

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

int main()
{
    double d = 123.456789123456;

    cout << d << '*' << endl;
    cout.precision(10);
    cout << d << '*' << endl;
    cout.precision(3);
    cout << d << '*' << endl;
    cout.precision(5);
    cout << d << '*' << endl;
    cout.precision(20);
    cout << d << '*' << endl;
    return 0;
}
```

运行结果:

```
123.457*
123.4567891*
123*
123.46*
123.45678912345600509*
```

结论:

- 1、不做任何设置的情况下, 浮点数默认为__**小数**__(小数/指数)方式; 不设precision的输出宽度默认为__**6**__
- 2、默认情况下, precision设定的宽度是__**全部数据**__(全部数据/小数部分)
- 3、宽度__**不包含**__(包含/不包含)小数点
- 4、如果宽度超过有效位数, 则__**可以**__(可以/不可以)显示, 超出有效位数__**不可信**__(可信/不可信)



§. 基础知识题 - 与cout有关的成员函数的基本使用

2. 用于字符输出控制的流成员函数

例9: cout.precision()

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

int main()
{
    double d = 123.456789123456;
    cout.setf(ios::fixed);
    cout << d << '*' << endl;
    cout.precision(10);
    cout << d << '*' << endl;
    cout.precision(3);
    cout << d << '*' << endl;
    cout.precision(5);
    cout << d << '*' << endl;
    cout.precision(20);
    cout << d << '*' << endl;
    return 0;
}
```

运行结果:

```
123.456789*
123.4567891235*
123.457*
123.45679*
123.45678912345600508615*
```

结论:

- 1、加ios::fixed后, precision默认的宽度为__6__, 设定的宽度是__小数部分__(全部数据/小数部分)
- 2、宽度__不包含__(包含/不包含)小数点
- 3、如果宽度超过有效位数, 则__可以__(可以/不可以)显示, 超出有效位数__不可信__(可信/不可信)



§. 基础知识题 - 与cout有关的成员函数的基本使用

2. 用于字符输出控制的流成员函数

例10: cout.precision()

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

int main()
{
    double d = 123.456789123456;
    cout.setf(ios::scientific);
    cout << d << '*' << endl;
    cout.precision(10);
    cout << d << '*' << endl;
    cout.precision(3);
    cout << d << '*' << endl;
    cout.precision(5);
    cout << d << '*' << endl;
    cout.precision(20);
    cout << d << '*' << endl;
    return 0;
}
```

运行结果:

```
1. 234568e+02*
1. 2345678912e+02*
1. 235e+02*
1. 23457e+02*
1. 23456789123456005086e+02*
```

结论:

- 1、加ios::scientific后, precision默认的宽度为6, 设定的宽度是小数部分 (全部数据/小数部分)
- 2、宽度不包含 (包含/不包含) 小数点
- 3、如果宽度超过有效位数, 则可以 (可以/不可以) 显示, 超出有效位数不可信 (可信/不可信)



§. 基础知识题 - 与cout有关的成员函数的基本使用

2. 用于字符输出控制的流成员函数

例11: cout.precision()

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

int main()
{
    double d = 123.456789123456;
    cout.setf(ios::fixed);
    cout.precision(10);
    cout << d << '*' << endl;

    cout.setf(ios::scientific);
    cout.precision(10);
    cout << d << '*' << endl;

    return 0;
}
```

运行结果:

```
123.4567891235*
0x1.edd3c08729a5fp+6*
```

结论:

先设ios::fixed后, 再设ios::scientific,
则输出显示__**错误**__(正确/错误)

本页需填写答案



§. 基础知识题 - 与cout有关的成员函数的基本使用

2. 用于字符输出控制的流成员函数

例12: cout.precision()

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

int main()
{
    double d = 123.456789123456;
    cout.setf(ios::scientific);
    cout.precision(10);
    cout << d << '*' << endl;

    cout.setf(ios::fixed);
    cout.precision(10);
    cout << d << '*' << endl;

    return 0;
}
```

运行结果:

```
1.2345678912e+02*
0x1.edd3c08729a5fp+6*
```

结论:

先设ios::scientific后, 再设ios::fixed,
则输出显示__错误__(正确/错误)



§. 基础知识题 - 与cout有关的成员函数的基本使用

2. 用于字符输出控制的流成员函数

例13: cout.precision()

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

int main()
{
    double d = 123.456789123456;
    cout.setf(ios::fixed);
    cout.precision(10);
    cout << d << '*' << endl;
    cout.unsetf(ios::fixed);
    //此处添句话, 需用cout. 函数名
    cout.setf(ios::scientific);
    cout.precision(10);
    cout << d << '*' << endl;

    return 0;
}
```

将程序补充完整, 得到期望的运行结果:

```
123.4567891235*
1.2345678912e+02*
```

提示: 回忆并参考第3章的作业

所用的cout. `_unsetf(ios::fixed);` 等价于
cout << `_resetiosflags(ios::fixed)`;



§. 基础知识题 - 与cout有关的成员函数的基本使用

2. 用于字符输出控制的流成员函数

例14: cout.precision()

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

int main()
{
    double d = 123.456789123456;
    cout.setf(ios::scientific);
    cout.precision(10);
    cout << d << '*' << endl;
    cout.unsetf(ios::scientific);
    //此处添句话, 需用cout. 函数名
    cout.setf(ios::fixed);
    cout.precision(10);
    cout << d << '*' << endl;

    return 0;
}
```

将程序补充完整, 得到期望的运行结果:

```
1.2345678912e+02*
123.4567891235*
```

提示: 回忆并参考第3章的作业

所用的cout. `_unsetf(ios::scientific);` 等价于
cout << `_resetiosflags(ios::fixed);`