cout 格式化输出

有些时候，我们希望输出的内容能够按照我们指定的格式进行输出。

例如，输出某个浮点数时，我们想保留小数点后面两位；输出某个整数时，希望它能按8个数字的宽度输出，当宽度不足时左边补0，等等。

这个时候，我们就需要使用cout的流操作算子（你也可以叫做格式控制符）

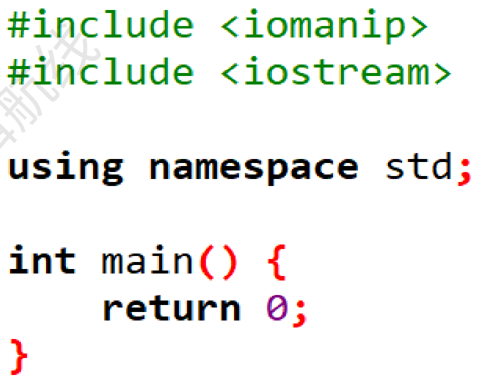
流操纵算子

dec	以十进制形式输出整数
hex	以十六进制形式输出整数
oct	以八进制形式输出整数
fixed	以普通小数形式输出浮点数
scientific	以科学计数法形式输出浮点数
left	左对齐，即在宽度不足时将填充字符添加到右边
right	右对齐，即在宽度不足时将填充字符添加到左边
setbase(b)	设置输出整数时的进制，b=8、10 或 16
setprecision(n)	我们可以通过setprecision指令，设定对当前数字保留的位数，注意，这里是按照四舍五入的方式进行保留。 setprecision配合fixed命令，可以实现保留指定位数的小数 setprecision配合scientific命令，可以实现保留指定位数的科学记数法
setw(w)	指定输出宽度为 w 个字符，或输人字符串时读入 w 个字符
setfill(c)	在指定输出宽度的情况下，输出的宽度不足时用字符 c 填充（默认情况是用空格填充）

代码示例

在编写cout的setiosflags算子时，我们需要引入iomanip和iostream头文件，有关头文件的知识，我们将在后面进行详解，在这里，我们只需要了解即可。

如下所示，在文件的最上方加上如下代码。



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

using namespace std;

int main() {
    return 0;
}
```

1、指定进制

```
cout << dec << 12 << "," << 24<<endl;
//使用十进制输出12,24
//输出结果：12,24
```

```
cout << hex << 12 << "," << 24<<endl;
//使用16进制输出12,24
//输出结果：c,18
```

```
cout << oct << 12 << "," << 24<<endl;
//使用8进制输出12,24
//输出结果：14,30
```

```
cout << setbase(16) << 12 << "," << 24<<endl;
//同 cout << hex << 12 << "," << 24;
//使用16进制输出12,24
//输出结果：c,18
```

2、小数的输出

```
cout << fixed << 12.12345678 << endl;
//以普通小数的形式进行输出，默认保留六位小数
//输出结果：12.123457
```

```
cout << scientific << 12.12345678 << endl;
//使用科学记数法输出小数
//输出结果：1.212346e+01
```

3、设置指定宽度及填充

```
cout << setw(12) << 100 << endl;
//设置输出宽度为12个字符，不足的地方用空白填充
//输出结果：         100
```

```
cout << setw(12) << setfill('0') << 100 << endl;
//设置输出宽度为12个字符，不足的地方用0来填充
//输出结果：000000000100
```

注意：每个setw仅对一个数字生效，如果想改变后续连续的输出宽度，则需要再次设置。例如：

```
cout << setw(8) << 12 << "," <<setw(8)<< 24<<endl;
//输出结果：      12,      24
```

4、设置数字的有效位数

我们可以通过setprecision指令，设定对当前数字保留的位数，注意，这里是按照四舍五入的方式进行保留，例如：

```
cout << setprecision(4) << 3.1415926<<endl;
//保留4位有效数字
//输出结果：3.142
```

setprecision配合fixed命令，可以实现保留指定位数的小数，例如：

```
cout << fixed << setprecision(4) << 3.1415926 << endl;
//保留4位小数
//输出结果：3.1416
```

setprecision配合scientific命令，可以实现保留指定位数的科学记数法，例如：

```
cout << scientific << setprecision(2) << 3.1415926 << endl;
//保留2位小数，使用科学记数法输出
//输出结果：3.14e+00
```

5、左对齐和右对齐

```
cout << setw(12) << right << 12.1 << endl;
//设置输出宽度为12个字符，使输出的数据保持右对齐
//输出结果：        12.1

cout << setw(12) << left << 12.1 << endl;
//设置输出宽度为12个字符，使输出的数据保持左对齐
//输出结果：12.1
```

6、换行

常用的换行有两种：endl和\n，如下所示：

```
cout << "We are" << endl;
cout << "in different\n";
cout << "line";
//输出结果：
//We are
//in different
//line
```

7、连字符

有些时候我们并不想写那么多的cout指令，但是又不想在同一行显示，有什么好办法呢？这个时候就需要用到连字符“\”，示例如下：

```
   cout << "We are in\
the same line";
   //使用连接符，将两行内容进行连接
  //输出结果：We are in the same line
```

练习题

1、在一行中右对齐输出3,4,5，每个数字占8个字符宽度，数字间用空格分割。

2、输出11.87243432，保留4位有效数字。

3、输出11.87243432，保留4位小数。

4、使用科学记数法输出11.87243432。

5、分三行输出字母a,b,c，要求是用两种不同的换行符。

6、分两行输出左对齐123和右对齐321，各占12个字符宽度。

7、分别用8,10,16进制输出10，各占一行。

8、分三行输出10,100,100，各占12个字符宽度，使用0填充空位。

9、使用连字符，在一行输出hello world。

参考代码

1、

```
cout << setw(8) << 3 << " " << setw(8) << 4 << " " << setw(8)
<< 5 << " " << endl;
```

2、

```
cout << setprecision(4) << 11.87243432;
```

3、

```
cout << fixed << setprecision(4) << 11.87243432;
```

4、

```
cout << scientific << 11.87243432;
```

5、

```
cout << "a" << endl;
cout << "b\n";
cout << "c";
```

6、

```
cout << left << setw(12) << 123 << endl;
cout << right << setw(12) << 321 << endl;
```

7、

```
cout << oct << 10 << endl;
cout << dec << 10 << endl;
cout << hex << 10 << endl;
```

8、

```
cout << setw(12) << setfill('0') << 10 << endl;
cout << setw(12) << setfill('0') << 100 << endl;
cout << setw(12) << setfill('0') << 1000 << endl;
```

9、

```
cout << "Hello \
world";
```

逻辑航线培优教育，信息学奥赛培训专家。

扫码添加作者获取更多内容。