C++20引入了一个格式字符串函数`std::format`

其中使用{}来做占位符

```
std::cout << std::format("Demo1 {}\n", "hello"); // Demo1 hello
std::cout << std::format("Demo2 {}+{}={}\n", 1, 2, 1 + 2); // Demo2 1+2=3
```

在{}加入位置参数调整输出顺序

```
std::cout << std::format("Demo3 {1} {2} {0}\n", 1, 2, 3); // Demo3 2 3 1
```

指定精度

```
std::cout << std::format("Demo4 {:.5}\n", std::numbers::pi); // Demo4 3.1416
```

为了方便编写实现如下辅助函数print,用于直接输出格式化后的字符串

```
template<typename...Args>
void print(const std::string_view str_fmt, Args &&...args) {
    std::cout << std::vformat(str_fmt, std::make_format_args(args...));
}
```

指定占用空间

```
print("Demo5 {:5}\n", 1); // Demo5     1
```

左对齐

```
print("Demo6{:<5}\n", 1); // Demo61
```

右对齐

```
print("Demo7{:>5}\n", 1); // Demo7    1
```

指定字符补齐

```
print("Demo8{:*>5}\n", 1); // Demo8****1
```

居中

```
print("Demo9{:^5}\n", 1); // Demo9  1
print("Demo10{:_^5}\n", 1); // Demo10__1__
```

二进制

```
print("Demo11 {:b}\n", 3); // Demo11 11
```

八进制

```
print("Demo12 {:o}\n", 17); // Demo12 21
```

十进制

```
print("Demo13 {:d}\n", 012);//Demo13 10
```

十六进制

```
print("Demo14 {:x} {:X}\n", 12, 12);//Demo14 c C 根据x的大小写来确定最后输出的大小写
```