# 第29章 字面量优化

《现代C++语言核心特性解析》 谢丙堃

# 十六进制浮点字面量

- std::hexfloat和std::defaultfloat

```
#include <iostream>
int main()
{
    double float_array[]{ 5.875, 1000, 0.117 };
    for (auto elem : float_array) {
        std::cout << std::hexfloat << elem
            << " = " << std::defaultfloat << elem << std::endl;
    }
}

/* output
    0x1.78p+2 = 5.875
    0x1.f4p+9 = 1000
    0x1.df3b645a1cac1p-4 = 0.117
*/
```

# 十六进制浮点字面量

- 源代码中书写十六进制浮点字面量

```
#include <iostream>
int main()
{
    double float_array[]{ 0x1.7p+2, 0x1.f4p+9, 0x1.df3b64p-4 };
    for (auto elem : float_array) {
        std::cout << std::hexfloat << elem
            << " = " << std::defaultfloat << elem << std::endl;
    }
}

/* output
    0x1.7p+2 = 5.75
    0x1.f4p+9 = 1000
    0x1.df3b64p-4 = 0.117
*/
```

# 二进制整数字面量

- 前缀0b和0B

```cpp
#include <iostream>
int main()
{
    auto x = 0b11001101L + 0xcdl + 077LL + 42;
    std::cout << "x = " << x << ", sizeof(x) = " << sizeof(x) << std::endl;

}

/* output
    x = 515, sizeof(x) = 8
*/
```

# 单引号作为整数分隔符

- 单引号整数分隔符对于十进制，八进制，十六进制，二进制整数都是有效的

```
constexpr int x = 123'456;
static_assert(x == 0x1e'240);
static_assert(x == 036'11'00);
static_assert(x == 0b11'110'001'001'000'000);
```

# 原生字符串字面量

- 普通字符串

```
char hello_world_html[] =
    "<!DOCTYPE html>\r\n"
    "<html lang = \"en\">\r\n"
    "  <head>\r\n"
    "  <meta charset = \"utf-8\">\r\n"
    "  <meta name = \"viewport\" content = \"width=device-width\">\r\n"
    "    <title>Hello World!</title>\r\n"
    "  </head>\r\n"
    "  <body>\r\n"
    "    Hello World!\r\n"
    "  </body>\r\n"
    "</html>\r\n";
```

# 原生字符串字面量

- 原生字符串

```
char hello_world_html[] = R"(<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width">
  <title>Hello World!</title>
</head>
<body>
Hello World!
</body>
</html>
)";
```

# 用户自定义字面量

- 基本语法

```
retrun_type operator "" identifier (params)
```

- 例子

```
#include <iostream>
long double operator"" _mm(long double x) { return x / 1000; }
long double operator"" _m(long double x)  { return x; }
long double operator"" _km(long double x) { return x * 1000; }
int main()
{
    std::cout << 1.0_mm << '\n';
    std::cout << 1.0_m  << '\n';
    std::cout << 1.0_km << '\n';
}
```

# 感谢您的观看

# 欢迎关注