



# USER-DEFINED LITERALS

Выполнила Старобыховская А.А. V3316



# Литералы

**Литерал** — это некоторое выражение, создающее объект.

Примеры:

- `'x'; // character`
- `"some"; // c-style string`
- `7.2f; // float`
- `74u; // unsigned int`
- `74l; // long`

# Классификация

- литералы-суффиксы (**121.45\_f**)
- литералы префиксы ( **0x234**)
- префиксо-суффиксные (**"124"**)

# Пользовательские литералы

- *сырые литералы (raw)*

литерал, который разбирает входное число посимвольно (число передается в оператор как строка).

- *литералы для встроенных типов (cooked).*

- Стандартная библиотека содержит литералы для *std::string*, *std::complex* и ед. измерения в операциях со временем в заголовке *<chrono>*.

```
std::string str = "hello"s + "World"s; // Standard Library <string> UDL
complex<double> num =
    (2.0 + 3.01i) * (5.0 + 4.3i); // Standard Library <complex> UDL
auto duration = 15ms + 42h; // Standard Library <chrono> UDLs
```

# Сигнатуры

- // INTEGRAL literal

**ReturnType operator "" \_a(unsigned long long int);**

- // FLOATING literal

**ReturnType operator "" \_b(long double);**

- // STRING literal

**ReturnType operator "" \_g(const char\*, size\_t);**

- // Raw literal operator

**ReturnType operator "" \_r(const char\*);**

- // Literal operator template

**template<char...> ReturnType operator "" \_t();**

# Сырые литералы

## ■ Сигнатура:

**OutputType operator "" \_suffix(const char\* literalString);**

## ■ Пример

**OutputType operator "" \_x(unsigned long long);**

**OutputType operator "" \_y(const char\*);**

**1234\_x;     // call: operator "" \_x(1234);**

**1234\_y;     // call: operator "" \_y("1234");**

# Литералы с обработкой

```
unsigned long long operator "" _min(unsigned long long minutes)
{
    return (minutes * 60);
}

// ...

std::cout << 5_min << std::endl;
```

***// на экран выведется 300***



- Если *определенный пользователем* литерал совпадает с *системным* (например f), то выполнится системный.

```
long operator "" f(long double value)
```

```
{
```

```
    return long(value);
```

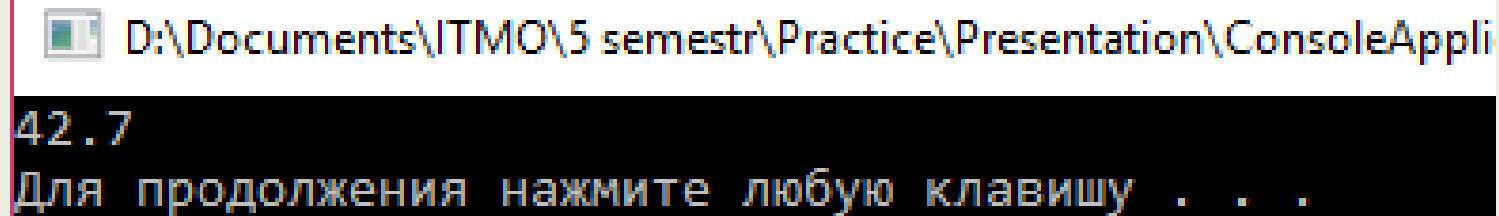
```
}
```

```
// ...
```

```
std::cout << 42.7f << std::endl;
```

Что выведет программа?

```
long operator "" f(long double value)  
{  
    return long(value);  
}  
// ...  
std::cout << 42.7f << std::endl;
```



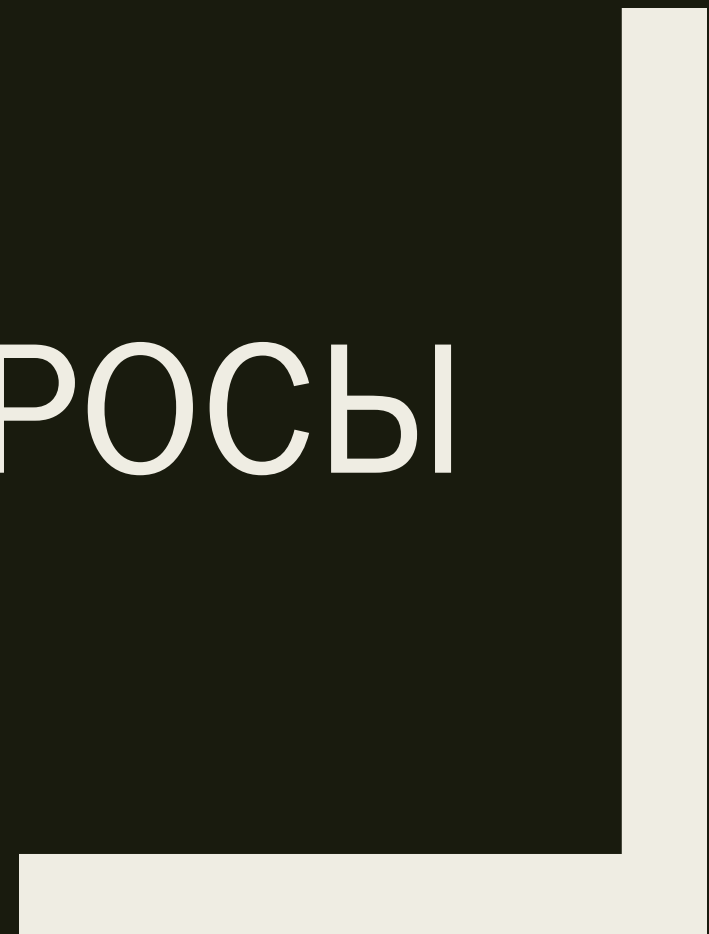
D:\Documents\ITMO\5 semestr\Practice\Presentation\ConsoleAppli

42.7  
Для продолжения нажмите любую клавишу . . .

# Поддержка компиляторами

- *gcc 4.7*
- *clang 3.1.*

ВОПРОСЫ



# Вопрос 1.

- Какие литералы можно создавать?

# Вопрос 1.

■ Какие литералы можно создавать?

*(C++ позволяет создавать только литералы-суффиксы)*

## Вопрос 2.

- Можно ли увеличить производительность кода за счет использования пользовательских литерал?

## Вопрос 2.

- Можно ли увеличить производительность кода за счет использования пользовательских литералов?

Определяемые пользователем литералы не дают выигрыша в производительности. Они служат, главным образом, для удобства и для определения типов во время компиляции



# Вопрос 3.

Что вернет программа?

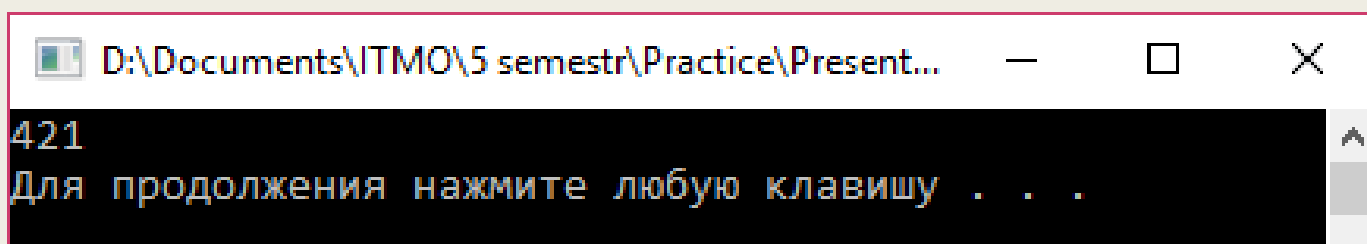
```
long long operator "" u(const long double num)
{
    return 412;
}
```

```
int main()
{
    std::cout << 421u << std::endl;
    system("pause");
    return 0;
}
```

# Вопрос 3.

```
long long operator "" u(const long double num)
{
    return 412;
}
```

```
int main()
{
    std::cout << 421u << std::endl;
    system("pause");
    return 0;
}
```



The screenshot shows a Windows command prompt window with the title bar "D:\Documents\ITMO\5 semestr\Practice\Present...". The window has standard Windows window controls (minimize, maximize, close). The command prompt displays the output "421" on the first line and "Для продолжения нажмите любую клавишу . . ." on the second line. A vertical scrollbar is visible on the right side of the command prompt window.

Спасибо за внимание!