



C++. Уровень 3

## Урок 5

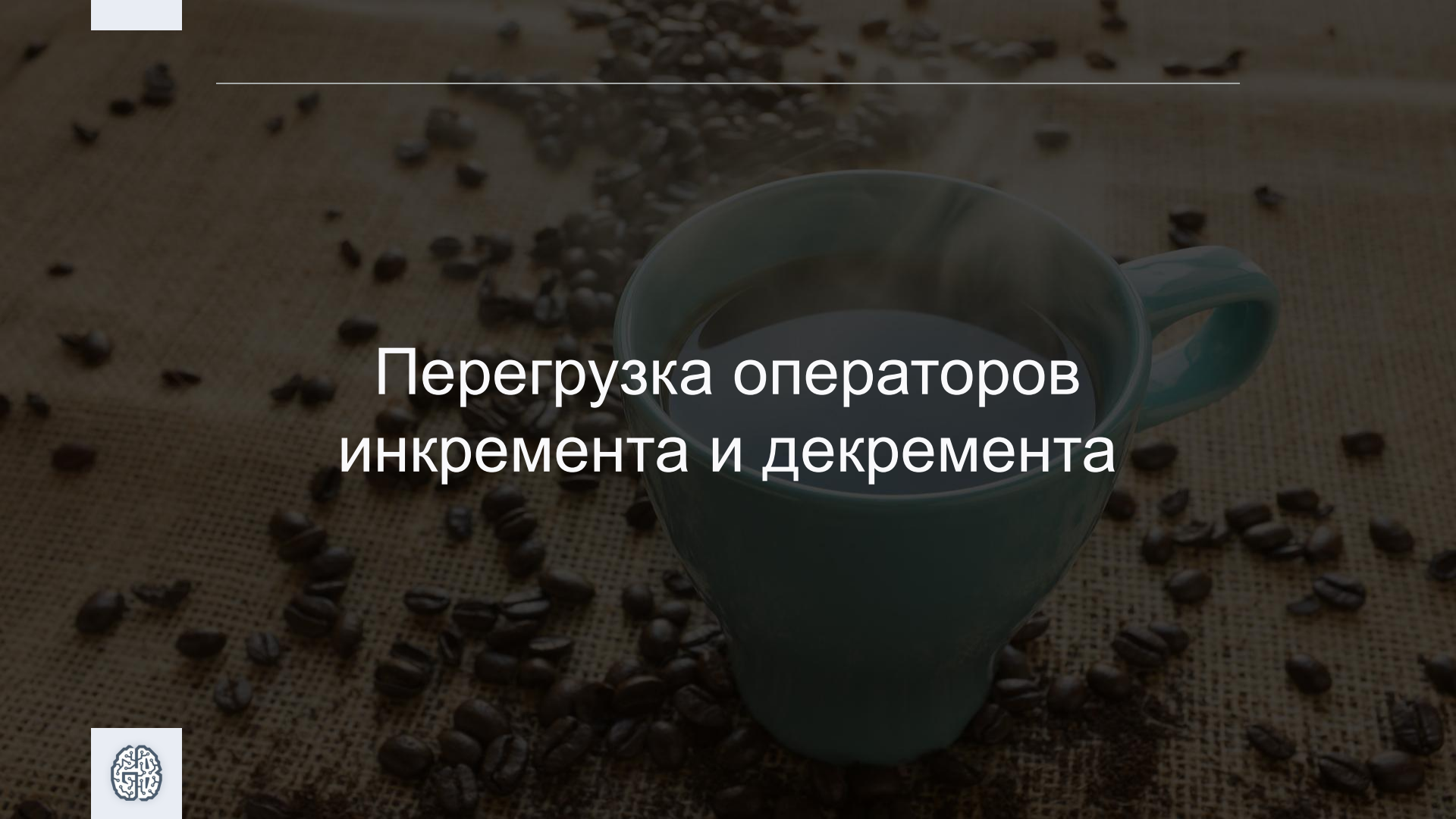
# Совместное использование функций и методов

Перегрузка операторов. Шаблоны  
функций и классов. Специализация  
шаблонов.

# План урока

- Перегрузка операторов инкремента и декремента.
- Шаблоны функций.
- Шаблоны классов.
- Явная специализация шаблона.
- Частичная специализация шаблона.



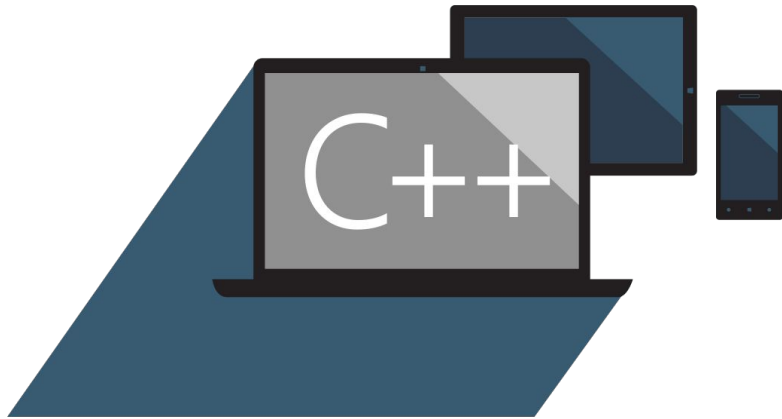
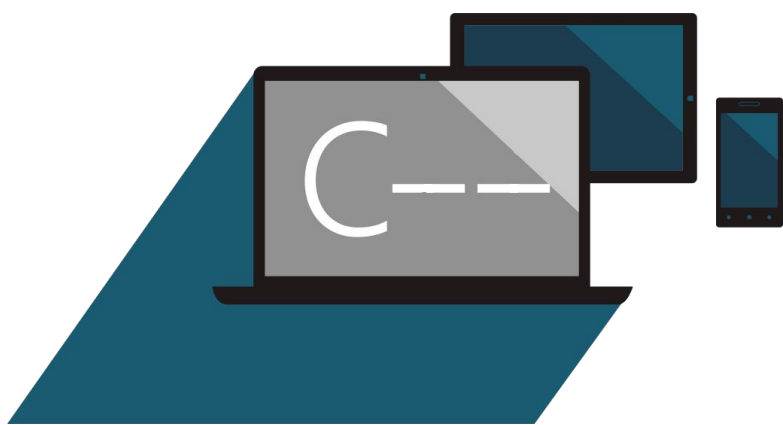


---

# Перегрузка операторов инкремента и декремента



# Операторы инкремента и декремента



# Перегрузка операторов:

- через методы класса;
- `operator++` как для версии префикс, так и для постфикс;
- версия постфикс реализуется через версию префикс;
- в версии постфикс создается «фиктивная» переменная.



**Рассмотрим код**



# Шаблоны функций и классов

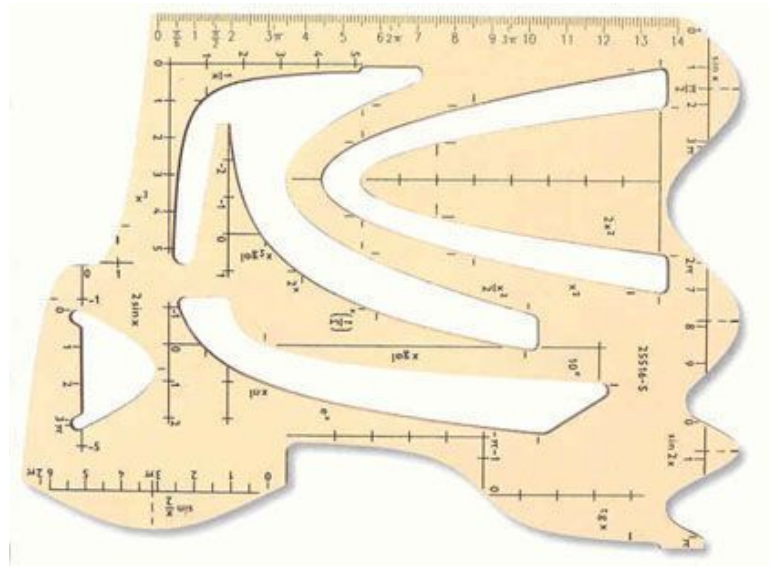




# Шаблоны функций

### Пример программы:

```
template <typename T>
T max(T a, T b)
{
    return (a > b) ? a : b;
}
```





# Экземпляры шаблона функции

```
int i = t_max(4, 8);  
double d = t_max(7.56, 21.434);  
char ch = t_max('b', '9');
```

```
int max(int a, int b)  
{  
    return (a > b) ? a : b;  
}
```

```
double max(double a, double b)  
{  
    return (a > b) ? a : b;  
}
```

```
char max(char a, char b)  
{  
    return (a > b) ? a : b;  
}
```



# Шаблоны классов

Шаблоны создаются для классов, имеющих  
общую логику работы

## Синтаксис шаблона класса:

```
template <список_параметров_шаблона_типа>  
class имя_класса  
{  
    // Реализация класса  
}
```

## Синтаксис экземпляра шаблона класса:

```
имя_класса <список_фактических_параметров> объект;
```



# Специализация шаблонов



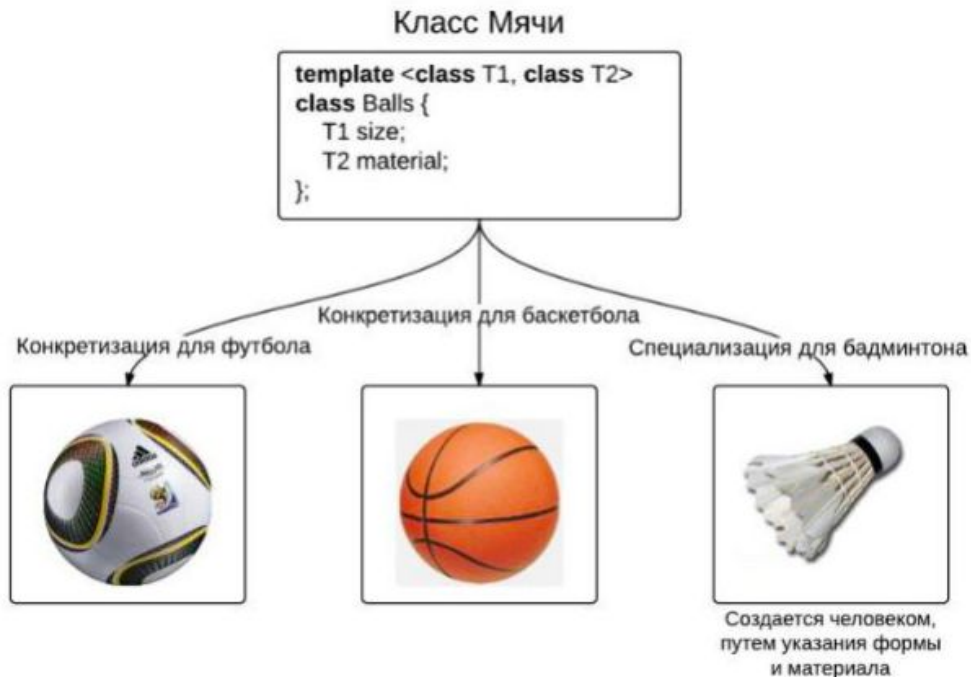
# Специализация шаблона функции

```
template <class T>
class Day
{
    // реализация шаблона
};

template <>
Day<char*>::Day(char* day)
{
    // код специализации для char*
}
```



# Специализация шаблона класса



Необходима, если шаблон классов не пригоден для конкретизации определенным типом данных или если конкретизация шаблона классов неэффективна по реализации.



# Частичная специализация шаблона класса

Проблема с полной специализацией шаблона заключается в том, что все параметры шаблона должны быть явно определены.

Частичная специализация шаблона позволяет выполнить специализацию шаблона класса (но не функции!), где некоторые, но не все, параметры шаблона явно определены.

```
template <typename T>  
class vector { /* ... */};
```

```
// полная специализация  
template<>  
class vector<bool> { /* ... */};
```

```
// частичная специализация (для подтипа)  
template <typename T>  
class vector<T*> { /* ... */};
```



---

# Решите задачи





Найдите ошибочные объявления (или пары объявлений) шаблонов классов:

*Продолжение*

1

```
(a)template <class Type> class Container1;  
template <class Type, int size>  
class Container1;
```

```
(b)template <class T, U, class V>  
class Container2;
```

```
(c)template <class C1, typename C2>  
class Container3 {};
```

```
(d)template <typename myT, class myT>  
class Container4 {};
```

```
(e)template <class Type, int *pi>  
class Container5;
```

```
(f) template <class Type, int val = 0>  
class Container6;  
template <class T = complex<double>, int v>  
class Container6;
```



2

Укажите, какие из данных конкретизированных шаблонов действительно приводят к конкретизации:

```
template < class Type > class Stack { };  
void f1( Stack< char > );           // (a)  
class Exercise {  
    //...  
    Stack< double > &rsd;           // (b)  
    Stack< int > si;                // (c)  
};  
  
int main() {  
    Stack< char > *sc;              // (d)  
    f1( *sc );                    // (e)  
    int iobj = sizeof( Stack<string > ); // (f)  
}
```



Какие из следующих конкретизаций шаблонов корректны?

```
template < int *ptr > class Ptr { ... };  
template < class Type, int size > class Fixed_Array { ... };  
template < int hi, int wid > class Screen { ... };
```

- 3
- (a) `const int size = 1024;`  
    `Ptr< &size>bp1;`
  - (b) `int arr[10];`  
    `Ptr< arr > bp2;`
  - (c) `Ptr < 0 > bp3;`
  - (d) `const int hi = 40; const int wi = 80;`  
    `Screen< hi, wi+32 > sObj;`
  - (e) `const int size_val = 1024;`  
    `Fixed_Array< string, size_val > fa1;`
  - (f) `unsigned int fasize = 255;`  
    `Fixed_Array< int, fasize > fa2;`
  - (g) `const double db = 3.1415;`  
    `Fixed_Array< double, db > fa3;`



# Вопросы участников

