OTUS

Шаблоны классов и функций

otus.ru

Проверить, идет ли запись

Меня хорошо видно && слышно?

Тема вебинара

Шаблоны классов и функций

Карина Дорожкина

Research Development Team Lead

Более 10 лет опыта разработки на C/C++.
Долгое время занималась развитием ПО в области безопасности транспортного сектора.
Имею опыт руководства несколькими командами и проектами с разнообразным стеком технологий.
Спикер конференций C++ Russia, escar Europe.
dorozhkinak@gmail.com

Правила вебинара

Активно
участвуем

Off-topic обсуждаем
в telegram #**C++-basic-2023-03**

Задаем вопрос в чат

Вопросы вижу в чате, могу
ответить не сразу

Мультипарадигменный C++

Как можем писать на C++?

- процедурное программирование
- объектно-ориентированное программирование
- **обобщенное программирование**
- метапрограммирование

Маршрут вебинара

- Обобщенное программирование
- Шаблоны функций
- Шаблоны классов
- Шаблоны переменных
- Заключение

Цели вебинара

После занятия вы сможете

1. Прочитать код с использованием шаблонов
2. Использовать шаблоны для обобщенного программирования
3. Правильно организовывать код с шаблонами

Смысл

Зачем вам это уметь

1. Шаблоны один из мощных инструментов языка C++
2. Многие библиотеки написаны с использованием шаблонов

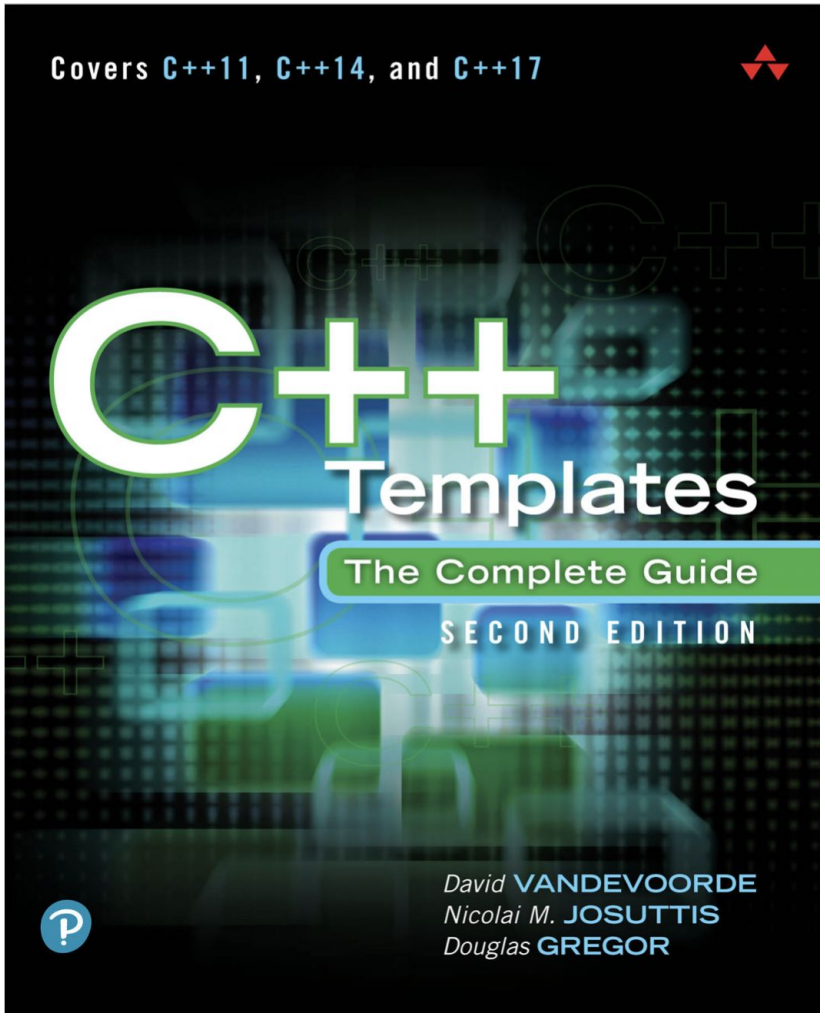
Обобщенное программирование

Шаблоны в C++

Обобщенное программирование (generic programming) - парадигма программирования, в которой описание данных и алгоритмов можно применить к различным типам данных.

Шаблоны (templates) - элемент языка C++ для написания кода в парадигме обобщенного программирования.

Шаблоны в C++


Covers C++11, C++14, and C++17
C++ Templates
The Complete Guide
SECOND EDITION
David VANDEVOORDE
Nicolai M. JOSUTTIS
Douglas GREGOR

Шаблоны функций

Шаблоны функций

main.cpp

```
#include <sum.h>
#include <func.h>

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

void main()
{
    std::cout << max(2, 3);
    std::cout << max(2.3f,
3.5f);
}
```

компилируем

main.S

```
__Z3maxIiET_S0_S0_:
…
    cmp  w1, w0
…

__Z3maxIfET_S0_S0_:
…
    fcmpe    s1, s0
…
```

Шаблоны функций

- можно инстанцировать для любых типов, поддерживающих требуемые операции
- компилятор может сам вывести из аргументов типы для инстанцирования шаблона функции
- можно специализировать реализацию под конкретные типы
- объявление, реализация и инстанциация шаблона должна быть в рамках одной единицы трансляции
- с помощью шаблонов можно выполнять вычисления на этапе компиляции (compile time programming)

Инстанцирование шаблона

Для обычных функций

sum.h

```
int sum(int a, int b);
```

sum.cpp

```
int sum(int a, int b)
{
    return a + b;
}
```

main.cpp

```
#include <sum.h>

void main()
{
    std::cout << sum(2, 3);
}
```

Для шаблонных функций

sum.h

```
template <typename T>
T sum(T a, T b);
```

sum.cpp

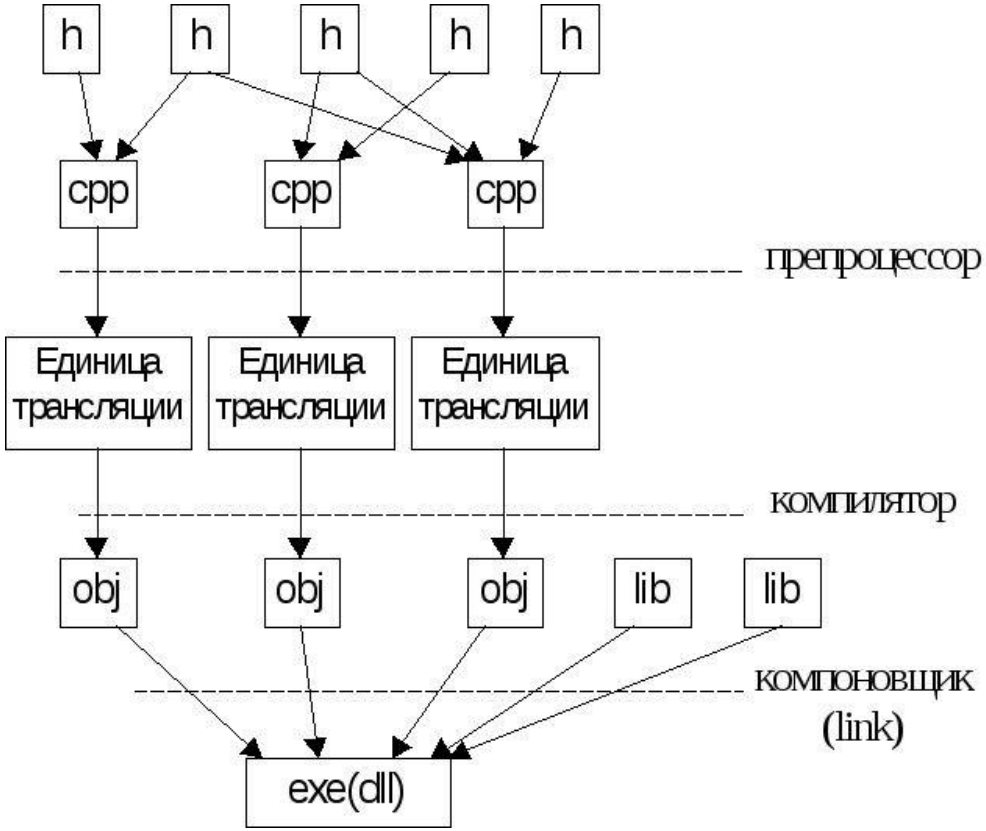
```
template<typename T>
T sum(T a, T b)
{
    return a + b;
}
```

main.cpp

```
#include <sum.h>

void main()
{
    std::cout << sum(2, 3);
}
```

Вспоминаем вебинар про компиляцию и линковку



Проблема linker undefined symbol

одна единица трансляции

sum.h

```
template <typename T>
T sum(T a, T b);
```

main.cpp

```
#include <sum.h>

void main()
{
    std::cout << sum(2, 3);
}
```

другая единица трансляции

sum.cpp

```
template<typename T>
T sum(T a, T b)
{
    return a + b;
}
```

Решение 1 вся реализация в одном .h

sum.h

```
template <typename T>
T sum(T a, T b)
{
    return a + b;
}
```

main.cpp

```
#include <sum.h>

void main()
{
    std::cout << sum(2, 3);
}
```

Решение 2

одна единица трансляции

sum.h

```
template <typename T>
T sum(T a, T b);

#include <sum.ipp>
```

sum.ipp

```
template<typename T>
T sum(T a, T b)
{
    return a + b;
}
```

main.cpp

```
#include <sum.h>

void main()
{
    std::cout << sum(2,
3);
}
```

Решение 3 explicit instantiation

одна единица трансляции

sum.h

```
template <typename T>
T sum(T a, T b);
```

main.cpp

```
#include <sum.h>

void main()
{
    std::cout << sum(2, 3);
}
```

другая единица трансляции

sum.cpp

```
template<typename T>
T sum(T a, T b)
{
    return a + b;
}

template int sum<int>(int a, int b);
```

Погодите...

sum.h

```
template <typename T>
T sum(T a, T b)
{
    return a + b;
}
```

main.cpp

```
#include <sum.h>
#include <func.h>

void main()
{
    func();
    std::cout << sum(2, 3);
}
```

func.cpp

```
#include <func.h>

void func()
{
    std::cout << sum(4, 3);
}
```

Эээ...

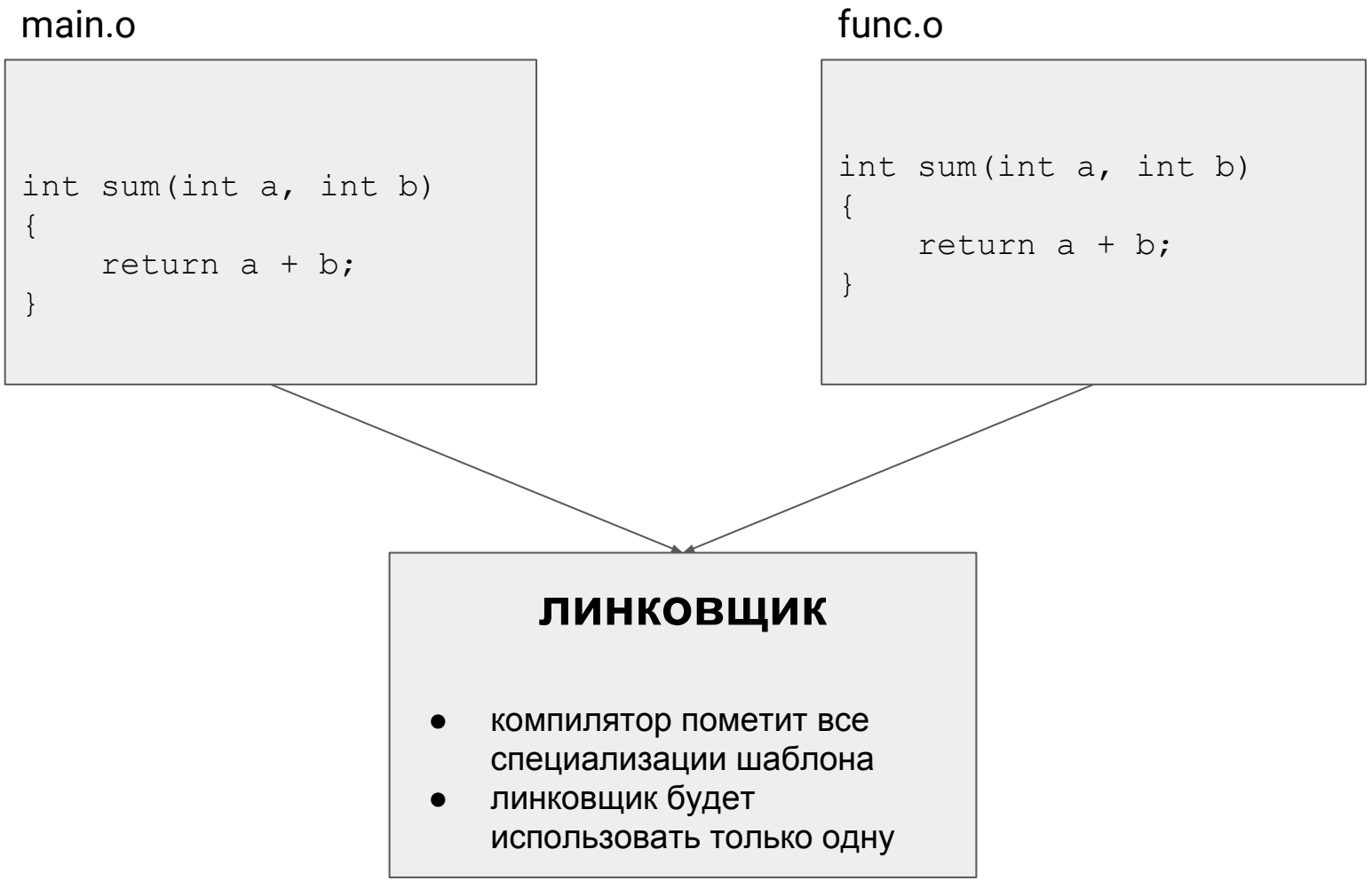
main.o

```
…
int sum(int a, int b)
{
    return a + b;
}
```

func.o

```
…
int sum(int a, int b)
{
    return a + b;
}
```

Жадное инстанцирование



Шаблоны классов

Шаблоны классов

- справедливы почти все утверждения как и для шаблонов функций
- шаблоны классов поддерживают частичную специализацию шаблона
- отдельные методы класса можно сделать шаблонными

Шаблоны переменных

Шаблоны переменных

```
template<class T>
T pi = T(3.14159265358979323851);

template<class T>
T circular_area(T r)
{
   return pi<T> * r * r;
}
```

Заключение

Ключевые тезисы

1. Шаблоны C++ - инструмент для обобщенного программирования
2. Шаблоны можно применять к функциям, классам и переменным
3. Объявление, реализация и инстанцирование шаблона должно быть в одной единице трансляции
4. Значения могут выступать параметрами шаблона

Цели вебинара

После занятия вы сможете

1. Прочитать код с использованием шаблонов
2. Использовать шаблоны для обобщенного программирования
3. Правильно организовывать код с шаблонами

Вопросы?

Заполните, пожалуйста, опрос о занятии по ссылке в чате