Шаблоны — механизм языка C++, который обеспечивает создание обобщенных алгоритмов. Так можно реализовать обобщенное поведение функции для разных встроенных типов. Например сложение для int, double.

Процесс подстановки(замены) шаблонного типа конкретным — инстанцирование.

Специализация шаблона — возможность определить шаблон индивидуально для конкретного типа. Бывает явной(полной) и частичной.

Явная используется для определения шаблонов функций.

Если есть нешаблонная(обычная) функция, которая идеально соответствует фактическим параметрам наравне с шаблонной, то предпочтение отдается нешаблонной.

Среди множества перегруженных функций(нешаблонных) компилятор подбирает ту, которая больше всех соответсвует передаваемым аргументам. При этом, если фактические и формальные параметры имеют разные типы, то компилятор производит преобразование типов (promotion):

- int, double(float) -> double(float)
- double, float -> double
- int, long int -> long int
- занковый, беззнаковый -> беззнаковый

Вместо преобразования типа выбирается шаблонный вариант(если таковой имеется)

```
#include <iostream>
using namespace std;


double sum(double a, double b) {
    std::cout << "0, ";
    return a + b;
}


template<typename T>
T sum(T a, T b) {
    std::cout << "1, ";
    return a + b;
}

int main()
{
    // template is used
    cout << sum(1.23f, 6.0f) << endl;
    return 0;
}
```

Использование шаблона с типом, который не поддерживает описанные в нем операции

```
#include <iostream>

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

struct S {
    int value;
};


int main()
{
    S a;
    a.value = 10;
    S b;
    b.value = 11;
    sum(a, b);
    return 0;
}
```

Если для подходят и явная специализация, и обобщенный шаблон, то выбирается первое.

```
#include <iostream>
using namespace std;

template<typename T>
T sum(T a, T b) {
    std::cout << "1, ";
    return a + b;
}

template<>
double sum(double a, double b) {
    std::cout << "2, ";
    return a + b;
}

int main()
{
    cout << sum(2.3, 4.5) << endl;
    return 0;
}
```

При перегрузке может возникнуть ситуация, когда больше 1 шаблона подходят для инстанцирования. При этом возникнет ошибка.

```
#include <iostream>
using namespace std;

template<typename T>
T sum(T a, T b) {
    std::cout << "1, ";
    return a + b;
}
```

```
template<>
double sum(double a, double b) {
    std::cout << "2, ";
    return a + b;
}

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

int main()
{
    cout << sum(2.3, 4.5) << endl;
    return 0;
}
```