3.22 Понятия шаблонов, инстанцирования шаблонов, специализации шаблонов. Синтаксис определения шаблонов классов и шаблонов функций. Синтаксис определения специализации шаблонов

Определение *Шаблоны* — это средство языка, предназначенное для написания кода без привязки к конкретному типу.
Определение *Инстанцирование шаблона* — процесс создания конкретного класса/функции/using из шаблона путем подстановки аргументов. Процесс инстанцирования шаблонов происходит между препроцессингом и компиляцией. Происходит с проверкой типов аргументов (совместимость по присваиванию, по приведению типов, по вызываемым методам). Для классов нужно явное инстанцирование
Определение *Специализации шаблонов* нужны для случаев, когда мы хотим, чтобы с данным набором типов данных функция вела себя по-другому (т.е. как-то специально). Специализации активно используются в type_traits.
Что можно определить, как шаблон? Функцию, класс, псевдоним, переменную, метод класса

```
template <typename T, typename U>
T maximum(const T& a, const U& b) {
    return a > b ? a : b;
}

template <typename T>
class vector {

};

// since c++11
template <typename T>
using mymap = std::map<T, T>;

// since c++14
template <typename T>
const T pi = 3.14;
```

```
template <typename T>
struct vector {
    template <typename U>
    void push_back(const U&);
};

template <typename T>
template <typename U>
void vector<T>::push_back(const U& x) {
    // ...
}
```

Пример специализации шаблонов для классов

```
template <typename T>
class vector {

};

// full specialization.
template <>
class vector<bool> {

};
```

Это полная специализация. Создаем частный случай шаблонного класса. При этом новый и старый классы могут вообще не совпадать в реализации.

```cpp
// partial specialization.
template <typename T>
struct S<T*> {
    void f() {
        std::cout << 3;
    }
};
```

Частичная специализация. определенная для всех типов, являющихся укзателями

Пример специализации шаблонов для функций

Для функций частичная специализация запрещена стандартом, можно делать только полную

```cpp
template <typename T, typename U>
void f(T, U) {
    std::cout << 1;
}

template <typename T>
void f(T, T) {
    std::cout << 2;
}

template <>
void f(int, int) {
    std::cout << 3;
}
```

Функция 3 - специализация 2 функции

```cpp
template <typename T, typename U>
void f(T, U) {
    std::cout << 1;
}

template <>
void f(int, int) {
    std::cout << 3;
}

template <typename T>
void f(T, T) {
    std::cout << 2;
}
```

Функция 3 - специализация 1 функции