Язык C++

Template specialization. Smart Ptr

Специализация шаблона класса

```
template<class T>
struct Boo {
   void foo() {
        std::cout << "foo" << std::endl;
   }
};

template<>
struct Boo<int> {
   void foo() {
      std::cout << "foo(int)" << std::endl;
   }
};
```

Full template specialization

1. Шаблон функции
2. Шаблон класса
3. Шаблон переменной
4. Шаблона функции класс
5. Шаблона члена класса
6. ….

Специализация шаблона класса

```
template<class T>
struct Boo {
    void foo() {
        std::cout << "foo" << std::endl;
    }
    void func () {};
};

template<>
struct Boo<int> {
    void foo() {
        std::cout << "foo(int)" << std::endl;
    }
};
```

Специализация шаблона класса

```
int main() {
    std::vector<bool> bv;
    std::vector<int> bi;
    return 0;
}
```

Специализация шаблона класса

```
template<class T>
struct is_float {
  static bool value() {  return false;  }
};

template<>
struct is_float<float> {
  static bool value() { return true; }
};

template<class T>
static bool is_float_v = is_float<T>::value();
```

Специализация шаблонов функций

```
template<class T>
void swap(T& a, T& b) {
  T tmp = a;
  a = b;
  b = tmp;
}

struct SomeStruct {};
```

Специализация шаблонов функций

```
template<>
void swap<SomeStruct>(SomeStruct& a, SomeStruct& b) {
  std::cout << "swap for SomeStruct with template" << std::endl;
}

void swap(SomeStruct& a, SomeStruct& b) {
  std::cout << "swap for SomeStruct without template" << std::endl;
}
```

Специализация шаблонов функций

```
template<class T>
void swap(std::vector<T>& x, std::vector<T>& y) {
  std::cout << "vector swap" << std::endl;
  x.swap(y);
};
```

Специализация шаблонного члена класса

```
struct SomeStruct {
  template<class T>
  void func(const T& x) {
      std::cout << x << std::endl;
  }

  void func(int x) {
      std::cout << "int" << std::endl;
  }
};
```

Частичная специализация

```
template<class T, class U>
struct Boo {
  void foo() { std::cout << "A" << std::endl; }
  void func () {};
};

template<class U>
struct Boo<int, U> {
  void foo() { std::cout << "B" << std::endl; }
};

template<>
struct Boo<int, int> {
  void foo() { std::cout << "C" << std::endl; }
};
```