Специализация и перегрузка шаблона.

- Специализация. Генерируется отдельный код для конкретного типа / типов.

Специализация шаблона в C++

↙ явная (полная)

определение спец. версии шаблона для опр-го типа

↘ частичная

в C++ предусм. для классов.

Перегрузка шабл. ф-ий.

- При явной конкретизации создается конкр. экземпляр. обобщ. шаблона.

- Когда нам требуется для нескольких типов опр-ть один шаблон, а для одного, самого конкретного, другой то исп-ся специализация.

```
template <typename T>
T f_name(T x) { }
....
template <>
f_name<type>(type x) { ... }
```

конкр. тип.

конкр. тип.

- Явная спец. должна быть указана раньше первого вызова неявной конкр. обобщ. шаблона.

Перегрузка шаблонных функций.

Обычная перегрузка ф-ий:

```
void whatami(int x) { ... }
void whatami(long x) { ... }
```

- Компилятор сам подбирает наилучшую подстановку ф-ии.
- Перегрузка нешаблонных ф-ий подразумевает перегрузку обычных

ф-ий, т.е. определение нескольких ф-ий с одинаковым именем, но различными параметрами. Т.к. набор пар-ов и их типы отличаются, то дублирование имени ф-ии допускается.

- Правило:
  1. Если сущ-ет нешабл. ф-ия, идеально соответ-ая факт. арг-ам, то она и выбирается.

     Иначе компилятор пробует конкретизировать все шаблоны ф-ий с тем же именем.
  - Если подходит и шабл. и нешабл., то выбирается нешабл. ф-ия
  - Шабл. ф-ия проверяется на соответствие, а не подгоняется под нужный тип.

Универсальные ссылки.
(Universal references)
Придумал Скотт Майерс.

```
void f(int& x);
void f(int&& x);
```

В данном случае будет работать перегрузка. Если будут rvalue, то мы попадаем в одну ф-ию, если lvalue - то в другую.
Если мы хотим, чтобы ф-ция могла по ссылке принимать оба вида value без исп. const (в данном случае знач-ие нельзя менять).

- При перегрузке шаблона выбор отдается в пользу более специфичного варианта.
- При наличии "очень" общих шаблонов подбирается точное соответствие. Нешабл. ф-ия, но с преобразо-

Кстати float → double преобразуется, если есть f88x.

- Более специфичный означает идеально точно соответствующий фактическому аргументу. Предпочтителен общему шаблону.

- Функции с переменным числом аргументов. В объявлении указывается многоточие. Например С-ф-ия printf.

```
void whatami(...) { ... }
```

Вызывается в самом последнем случае, если другие варианты не подошли.

- В отличие от неявных ф-ий в перегр. шаблонах (избегается неявное преобр-ие) предпочтение отдается варианту без преобразования. Т.е. в процессе разрешен смысл быть вариант перегруженной шаблон, а не шаблонная ф-ия.