Дз 10

1. *Ключевые слова auto, decltype и статическая типизация.*

`auto` — Выводит тип объявленной переменной из выражения инициализации.

`decltype` — это оператор, который позволяет определить тип выражения или переменной на этапе компиляции. Он возвращает тип выражения в качестве его результата, не выполняя его фактически.

Статическая типизация — это метод определения типов данных во время компиляции, когда все типы данных должны быть известны заранее и не могут быть изменены в дальнейшем. Это позволяет обнаруживать ошибки типов до выполнения программы.

2. *Инстанцирование шаблона: что это, как происходит.*

Инстанцирование шаблона — это процесс создания конкретной реализации шаблонного класса или функции на основе переданных аргументов типа. Истанцирование происходит во время компиляции.

Когда компилятор встречает вызов шаблонной функции или создание объекта шаблонного класса, он генерирует код для каждого использования шаблона с учетом переданных аргументов типа.

Различают неявное инстанцирование (происходит при вызове функции или создании объекта класса) и явное инстанцирование с помощью `template`.

3. *Зачем нужны шаблоны с нетиповыми аргументами, приведите примеры.*

Шаблоны с нетиповыми аргументами в C++ позволяют создавать обобщенные классы и функции, которые могут работать с различными типами данных и значениями, не зависящими от типа.

Примеры шаблонов с нетиповыми аргументами:

1. `std::array` — массив фиксированного размера, который может содержать элементы любого типа.

2. `std::max` — возвращает наибольшее значение из двух переданных аргументов, которые могут быть любого типа (тип аргументов определяется автоматически).
3. `std::pair` — пара значений, которые могут быть любого типа (типы значений определяются в качестве нетиповых аргументов шаблона)

4. *SFINAE*

"Substitution Failure Is Not An Error” — при определении перегрузок функции ошибочные инстанциации шаблонов не вызывают ошибку компиляции, а отбрасываются из списка кандидатов на наиболее подходящую перегрузку.

5. *Универсальные ссылки. std::forward.*

Универсальные ссылки (universal references) — это ссылки, которые могут принимать как lvalue-ссылки, так и rvalue-ссылки. Они объявляются с помощью оператора `&&` после типа данных, например:

```
//1
template <typename T>
void foo(T&& arg);

//2
Widget&& var1 = someWidget;
auto&& var2 = var1;
```

Когда функции передается lvalue, T будет заменен на тип lvalue-ссылки. Если функции передается rvalue, T будет заменен на тип значения, аргументу будет присвоено временное rvalue-значение.

`std::forward` используется для передачи универсальных ссылок в другие функции. Он позволяет сохранить тип ссылки и передать ее дальше, не теряя информации о том, был ли аргумент передан как lvalue или rvalue. Он принимает один аргумент - универсальную ссылку - и возвращает ее с сохранением типа: