Практическое занятие 3.17

Преобразование типов

Преобразование типов: static_cast<T>

Приводит один тип к другому. Если это примитивные типы, то будут использованы встроенные правила их приведения. Если это собственные классы, будут использованы правила приведения (*переопределение оператора приведения: operator Point() { …}*).
static_cast между указателями корректен, только если один из указателей – **void*** или если это приведение между объектами классов одной иерархии.

```
int* p = static_cast<int*>(malloc(100));
```

Если приведение не удалось, возникнет ошибка на этапе компиляции. Однако, если это приведение между указателями на объекты классов вниз по иерархии и оно не удалось, результат операции не определен.

Преобразование типов: dynamic_cast<T>

Безопасное приведение по иерархии наследования.

```
dynamic_cast<derv_class *>(base_class_ptr_expr)
```

Используется RTTI (Runtime Type Information), чтобы привести один указатель на объект класса к другому указателю на объект класса. Классы должны входить в одну иерархию. Если это условие не соблюдено, ошибка возникнет на этапе компиляции. Если приведение невозможно, то на этапе выполнения программы будет возвращен nullptr.

```
dynamic_cast<derv_class &>(base_class_ref_expr)
```

Работа со ссылками происходит почти как с указателями, но в случае ошибки во время исполнения будет выброшено исключение bad_cast.

Преобразование типов: reinterpret_cast<T>

Результат может быть некорректным, проверок не делается. Не может быть приведено одно значение к другому значению. Обычно используется, чтобы привести указатель к указателю, указатель к целому, целое к указателю. Умеет также работать со ссылками.

```
reinterpret_cast<whatever *>(some *)
reinterpret_cast<integer_expression>(some *)
reinterpret_cast<smth *>(integer_expression)
```

Чтобы использовать reinterpret_cast нужны очень и очень веские причины. Используется, например, при приведении указателей на функции.

Преобразование типов: const_cast<T>

Убирает спецификаторы const и volatile. Если приведение типов не удалось, выдается ошибка на этапе компиляции.
При использовании остальных приведений типов (не указателей) спецификаторы не меняются.

```
int a = 10;
const int* pointer1 = &a;
int* pointer2 = const_cast<int*>(pointer1);
```