

# Перечисления

```
enum Scale
{
    Kelvin,
    Celsius,
    Fahrenheit
};
```

```
Temperature data;
data.value = 100;
data.scale = Kelvin;
```

```
struct Temperature
{
    double value;
    Scale scale;
};
```

~~data.scale = 'Q';~~

✓ Так и нужно.

```
char symbol = data.scale;
cin >> data.scale;
```

× Возможность утрачена.

```
cout << data.scale; // 0
```

× Не этого мы хотели!

# Преобразование перечислений

## Из символа в шкалу

```
Scale to_scale (char symbol)
{
    switch (symbol) {
        case 'C': return Celsius;
        // ...
    }
}
```

```
char input;
cin >> input;
Scale value = to_scale(input);
```

## Из шкалы в символ

```
char from_scale (Scale value)
{
    switch (value) {
        case Celsius: return 'C';
        // ...
    }
}
```

```
Scale value = Fahrenheit;
cout << from_scale (value);
```

# Приведение типов (type cast)

- ✓ Значения для Kelvin, Celsius и т. д. разные,
- ✗ но неизвестные.

Сделаем их известными и удобными.

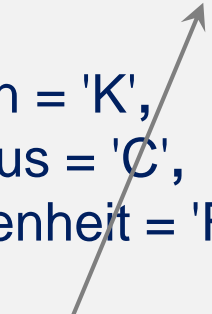
- Любые значения хранятся в памяти;
  - для `Scale` там коды символов `char`.
- На время вывода значения типа `Scale` можно и нужно рассмотреть как `char`.

```
Scale value = Celsius;  
cout << (char) value;
```

```
cin >> (char &) value;
```

- При вводе рассмотреть переменную типа `Scale` как переменную-`char`.

```
enum Scale : char  
{  
    Kelvin = 'K',  
    Celsius = 'C',  
    Fahrenheit = 'F'  
};
```



Нижележащий тип  
(underlying type):  
элементы `Scale` будут  
константами типа `char`.