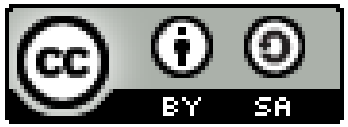
Перечисления и объединения

(**enum** and **union**)



May 25, 2016

Проблема

Использовать магические числа – плохо

Примеры использования магических чисел.

```
if( command == 42 ) {
  kill_all_users();
}

unsigned bits = 1024 * 768;

switch( code ) {
  case 0: go_left(); break;
  case 1: go_right(); break;
  case 2: full_stop(); break;
  case 3: run_forward(); break;
}
```

Решение проблемы методами C

```
#define CMD_KILL 42
// .....
if( command == CMD_KILL ) {
  kill_all_users();
}

#define N_LINES 768;   // so error here
#define PIXELS_PER_LINE 1024
unsigned bits = N_LINES * PIXELS_PER_LINE;

#define C_LEFT 0
#define C_RIGHT 1
// ....

switch( code ) {
  case C_LEFT:  go_left(); break;
  case C_RIGHT: go_right(); break;
  // ................
}
```

Достоинства и недостатки подхода в стиле C

Достоинства:

- избавляется от магических чисел.

Недостатки:

- используется препроцессор, который ничего не знает о языке;
- нет возможности контролировать типы;
- define можно переопределить;
- появляются трудноуловимые ошибки;
- компилятор не имеет возможности подсказать причину ошибки – он её не видит.

Часть проблем в C++ решается с помощью **const** и **constexpr**.

Решение проблемы методами C++
enum

```
enum Commands { // новый тип – перечисление
   stop = 0,
   run, // 1 = previous + 1
   left,
   right,
   fire = 100
};

Commands cmd = getCommand( commander )
switch( cmd ) {
   case stop: full_stop(); break;
   case left: go_left();   break;
   // ....
};
int a = cmd; // так можно но( не class enum)
cnd = 7;     // а так нельзя
```

Enum автоматически преобразуются в соответствующий целочисленный тип, **но не наоборот** !

Если несколько объектов надо расположить в одной и той же области памяти, то можно использовать union:

```
union EvilHack { // новый тип – объединение
  int i;
  char c[sizeof(int)];
  int *pi;
};

EvilHack a;
a.i = 65;
cout << '{' << c[0] << "} {" << c[3] << "}" << endl;

a.pi = nullptr;
cout << "a.i= " a.i << endl;
```

Использование: определение архитектурно-зависимых вещей и их использование, хаки и извращения.

enum class

Проблема: перечисления засоряют засоряют текущую область видимости своими именами, а также имеют неявное преобразование в интегральные типы.

```
void f() {
  enum E0 { a1, a2, a3, right };
  enum E1 { x1, x2, x3, right }; // error
  enum { go, stop };             // anonymous enum
   int a = right;
}
```

Решение (C++11): **enum class** ;

```
void f() {
  enum class E0 { a1, a2, a3, right };
  enum class E1 { x1, x2, x3, right };
   auto a = E0::right;
}
```

Не автоматического приведения к целочисленным типам. Указание типа в литерале обязательно.

Вложенные типы

Как и остальные пользовательские типы, перечисления и объединения могут быть вложены (принадлежать) классам и структурам.

```
class A {
  public:
    enum E { left, right, top, bottom };
};

class B {
  public:
    enum E { left = 50, right, top, bottom, inside };
};

auto v1 = A::E::top;
B::E v2 = B::right; // or B2::E::right;
```