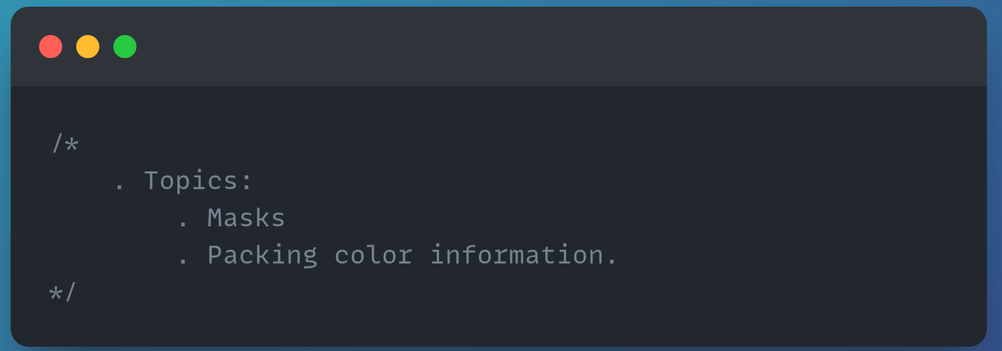
Bitwise operators



Flag parameters

```
export void use_options_v0(bool flag0, bool flag1, bool flag2, bool flag3, bool flag4, bool flag5, bool flag6, bool flag7)
{
  fmt::println("Flag0 is : {}, do something with it.", flag0);
  fmt::println("Flag1 is : {}, do something with it.", flag1);
  fmt::println("Flag2 is : {}, do something with it.", flag2);
  fmt::println("Flag3 is : {}, do something with it.", flag3);
  fmt::println("Flag4 is : {}, do something with it.", flag4);
  fmt::println("Flag5 is : {}, do something with it.", flag5);
  fmt::println("Flag6 is : {}, do something with it.", flag6);
  fmt::println("Flag7 is : {}, do something with it.", flag7);
}

export void use_options_v1(unsigned char flags)
{

  fmt::println("bit0 is : {}, do something with it!", ((flags & mask_bit_0) >> 0));
  fmt::println("bit1 is: {}, do something with it! ", ((flags & mask_bit_1) >> 1));
  fmt::println("bit2 is: {}, do something with it! ", ((flags & mask_bit_2) >> 2));
  fmt::println("bit3 is: {}, do something with it! ", ((flags & mask_bit_3) >> 3));
  fmt::println("bit4 is: {}, do something with it! ", ((flags & mask_bit_4) >> 4));
  fmt::println("bit5 is: {}, do something with it! ", ((flags & mask_bit_5) >> 5));
  fmt::println("bit6 is: {}, do something with it! ", ((flags & mask_bit_6) >> 6));
  fmt::println("bit7 is: {}, do something with it! ", ((flags & mask_bit_7) >> 7));
}
```

Packing colors

```
export void pack_colors(){
  constexpr unsigned int red_mask{ 0xFF000000 };
  constexpr unsigned int green_mask{ 0x00FF0000 };
  constexpr unsigned int blue_mask{ 0x0000FF00 };
  constexpr unsigned int alpha_mask{ 0x000000FF };// Transparency information


  unsigned int my_color{ 0xAABCDE00 };

  // We shift to make sure the color byte of interest is in the
  //  lower index byte position so that we can interpret that as an integer,
  //  which will be between 0 and 255.


  fmt::println("Red is : {0:d}", ((my_color & red_mask) >> 24));
  fmt::println("Green is : {0:d}", ((my_color & green_mask) >> 16));
  fmt::println("Blue is : {0:d}", ((my_color & blue_mask) >> 8));
  fmt::println("Alpha is : {0:d}", ((my_color & alpha_mask) >> 0));
}
```