Explicit object parameters (C++23)

```
/*
      . Topics:
          . Reasons:
              . Makes it possible to use recursive lambdas
              . Other advanced uses
                  . Generic Programming
          . Syntax on a Pixel class
          . Revisiting recursive lambdas


*/
```

Explicit object parameters (C++23)

```
export class Pixel {

   public:
       uint32_t get_color() const { return m_color; }

       void set_color(this Pixel& self, uint32_t color) {
          self.m_color = color;
       }

       unsigned int get_x() const {...}
       void set_x(unsigned int x) {...}

       unsigned int get_y() const {...}

       void print_access_count() const {
           fmt::print("Access count: {}\n", m_access_count);
       }

   private:
       uint32_t m_color{0xFF000000};
       unsigned int m_pos_x{0};
       unsigned int m_pos_y{0};
};
```

Recursive lambdas

```cpp
// Recursive lambda with explicit object parameters
auto fib = [](auto self, int n) -> int {
  if(n <= 1) return n;
  return self(self, n - 1) + self(self, n - 2);
};

fmt::println("Fibonacci(10): {}", fib(fib, 10));
```

Constructor initializer lists: inside the class definition

```
export class Pixel {
    public:
        Pixel();

        Pixel(uint32_t color, unsigned int x, unsigned int y)
            : m_color{color},
            m_pos_x{x},
            m_pos_y{y}
        {
            fmt::print("Pixel created\n");
        }

        uint32_t get_color() const;
        void set_color(uint32_t color);

        unsigned int get_x() const;
        void set_x(unsigned int x);

        unsigned int get_y() const;
        void set_y(unsigned int y);

    private:
        uint32_t m_color{0xFF000000};
        unsigned int m_pos_x{0};
        unsigned int m_pos_y{0};
};
```

Constructor initializer lists: outside

```
export class Pixel {
    public:

        //Declaration
        Pixel(uint32_t color, unsigned int x, unsigned int y);

        uint32_t get_color() const;
        void set_color(uint32_t color);

        unsigned int get_x() const;
        void set_x(unsigned int x);

        unsigned int get_y() const;
        void set_y(unsigned int y);

    private:
        uint32_t m_color{0xFF000000};
        unsigned int m_pos_x{0};
        unsigned int m_pos_y{0};
};

//Definition
Pixel::Pixel(uint32_t color, unsigned int x, unsigned int y) :
    m_color{color},
    m_pos_x{x},
    m_pos_y{y} {
    fmt::print("Pixel created\n");
}
```

Constructor initializer lists: Sometimes they are mandatory

```
export class Pixel {
  public:
      Pixel(uint32_t color, const Position& pos, const uint32_t& ref_color);  // Custom constructor

      uint32_t get_color() const;
      void set_color(uint32_t color);

      unsigned int get_x() const;
      void set_x(unsigned int x);

      unsigned int get_y() const;
      void set_y(unsigned int y);

  private:
      const uint32_t m_const_color;        // 1. Const member
      const uint32_t& m_ref_color;         // 2. Reference member
      Position m_position;                 // 3. Object without default constructor
      uint32_t m_color{0xFF000000};
      unsigned int m_pos_x{0};
      unsigned int m_pos_y{0};
};
```

Constructor initializer lists: Sometimes they are mandatory

```cpp
Pixel::Pixel(uint32_t color, const Position& pos, const uint32_t& ref_color)
    : m_const_color{color},
    m_ref_color{ref_color},
    m_position{pos.x, pos.y},
    m_color{color},
    m_pos_x{0},
    m_pos_y{0}
{
    fmt::print(
        "Pixel created with color: 0x{:08X} at Position({}, {}) using a reference to 0x{:08X}\n",
        color,
        pos.x,
        pos.y,
        ref_color
    );
}
```