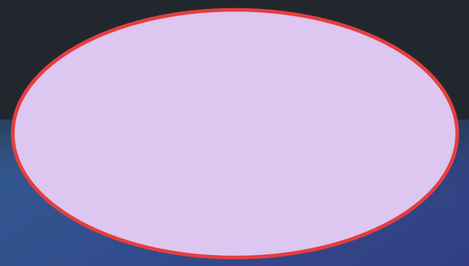
Custom Types

```
/*
      . Topics:
          . Modeling and instantiating types
          . Access specifiers
          . Constructors
          . Destructors
          . Setters and getters
          . Allocating objects
            . stack
            . heap
            . heap with smart pointers
          . Separating declarations from implementations
          . Explicity object parameters (C++23)
          . Constructor initializer lists
          . Copy semantics
          . Constructor delegation
          . Copy assignment operator
          . Reading and writting objects from/to files
          . Friends


*/
```



Creating a class

```
export module ct1;

namespace ct1{

    //export class Pixel {
    export struct Pixel {

        //public:
            uint32_t m_color{0xFF000000};
            unsigned int pos_x{0};
            unsigned int pos_y{0};
    };

} // namespace ct1
```

Flagging members private

```
export class Pixel {

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

};
```

Getters and setters

```
export class Pixel {
    public:
        uint32_t get_color() {
            return m_color;
        }
        void set_color(uint32_t color) {
            m_color = color;
        }

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

};
```

Const members

```
export class Pixel {
    public:
        uint32_t get_color() const {
            return m_color;
        }
        void set_color(uint32_t color) {
            m_color = color;
        }

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

};
```

Constructors

```
ct2::Pixel p{0x00FF00FF, 10, 20 };
```

Constructors

```
export class Pixel {
    public:
        //Consturctor with empty parameter list
        Pixel(){
        }
        //Constructor with parameters
        Pixel(uint32_t color, unsigned int x, unsigned int y){
            m_color = color;
            m_pos_x = x;
            m_pos_y = y;
        }
        //Destructor
        ~Pixel(){
            fmt::print("Pixel destroyed\n");
        }

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

};
```

Explicitly defaulted default constructor

```
export class Pixel {
    public:
        //Consturctor with empty parameter list
        Pixel() = default;
        //Constructor with parameters
        Pixel(uint32_t color, unsigned int x, unsigned int y){
            m_color = color;
            m_pos_x = x;
            m_pos_y = y;
        }
        //Destructor
        ~Pixel(){
            fmt::print("Pixel destroyed\n");
        }

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

};
```

Explicitly deleted default constructor

```
export class Pixel {
    public:
        //Consturctor with empty parameter list
        Pixel() = delete;
        //Constructor with parameters
        Pixel(uint32_t color, unsigned int x, unsigned int y){
            m_color = color;
            m_pos_x = x;
            m_pos_y = y;
        }
        //Destructor
        ~Pixel(){
            fmt::print("Pixel destroyed\n");
        }

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

};
```

The this keyword

```
export class Pixel {
    public:
        uint32_t get_color() const {
            return m_color;
        }
        void set_color(uint32_t color) {
            this→m_color = color;
        }

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

};
```

Different ways to create objects

```cpp
//Create the object on the stack
ct2::Pixel p{0x00FF00FF, 10, 20 };

//Change the data inside the Pixel object
p.set_color(0x00FF00FF);
p.set_x(10);
p.set_y(20);

fmt::println("Pixel color: {:#010x}", p.get_color());
fmt::println("Pixel position: ({}, {})", p.get_x(), p.get_y());

//Create objects on the heap with raw pointers
ct2::Pixel* p1 = new ct2::Pixel(0x00FF00FF, 10, 20);
fmt::println("Pixel color: {:#010x}", p1->get_color());
fmt::println("Pixel position: ({}, {})", p1->get_x(), p1->get_y());
delete p1;

//Create objects on the heap with smart pointers
auto p2 = std::make_unique<ct2::Pixel>(0x00FF00FF, 10, 20);
fmt::println("Pixel color: {:#010x}", p2->get_color());
fmt::println("Pixel position: ({}, {})", p2->get_x(), p2->get_y());
p2->print_access_count();
```

When the default constructor is needed

```
//When the default constructor is needed
ct2::Pixel p;    // Compilation error
ct2::Pixel pixels1[10];  // Compilation error
ct2::Pixel* pixels2 = new ct2::Pixel[10];  // Compilation error
ct2::Pixel pixels3[]{ct2::Pixel(0x00FF00FF, 10, 20), ct2::Pixel(0x00FF00FF, 10, 20)};  // OK
```

Mutable members

```
export class Pixel {

    public:
        Pixel() = default;  //Explicitly defaulted default constructor
        uint32_t get_color() const {
            ++m_access_count;
            return m_color;
        }

        unsigned int get_x() const {
            ++m_access_count;
            return m_pos_x;
        }
        unsigned int get_y() const {
            ++m_access_count;
            return m_pos_y;
        }
        void set_y(unsigned int y) {
            m_pos_y = y;
        }

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

    private:
        uint32_t m_color{0xFF000000};
        mutable unsigned int m_access_count{0}; // Tracks how many times the pixel has been accessed
};
```