Lambda functions

```
/*
    . Lambda functions:
        . Declaring a lambda function and calling it through a name
        . Declare a lambda function and call it directly
        . Lambda function that takes parameters
        . Lambda function that returns something
        . Explicitly specify the return type
        . Capture lists

*/
```

Lambda functions

```
/*
Lambda function signature :
                            [capture list] (parameters) →return type{
                                    // Function body
                            }
*/

// Declaring a lambda function and calling it through a name
auto func = [](){
    fmt::println( "Hello World!" );
};
func();
func();
```

Lambda functions: Declare and call

```
// Declare a lambda function and call it directly
[](){
  fmt::println( "Hello World!" );
}();
```

Lambda functions: Taking parameters

```
// Lambda function that takes parameters
[](double a, double b){
  fmt::println( "a + b : " , (a + b)  );
}(10.0,50.0);
```

Lambda functions: return values

```
// Lambda function that returns something
auto result = [](double a, double b){
  return a + b;
}(10,60);

//fmt::println( "result : " , result );
fmt::println( "result : " , [](double a, double b){
  return a + b;
}(10,60) );
```

Lambda functions: specify return type

```cpp
// Explicitly specify the return type
auto func3 = [](double a, double b) -> int { return a + b; };

auto func4 = [](double a, double b) { return a + b; };

double a{ 6.9 };
double b{ 3.5 };

auto result3 = func3(a, b);
auto result4 = func4(a, b);

fmt::println("result3 : {}", result3);
fmt::println("result4 : {}", result4);
fmt::println("sizeof(result3) : {}", sizeof(result3));// 4
fmt::println("sizeof(result4) : {}", sizeof(result4));// 8
```

Lambda functions: capture lists

```
// Capture lists

double a{10};
double b{20};

auto func = [a,b](){
  fmt::println( "a + b : " , a + b );
};
func();
```

Lambda functions: capture by value

```cpp
//Capturing by value
int c{42};

// Lambda capturing 'c' by value
auto func = [c]() {
  fmt::println("Inner value: {} &inner: {}", c, fmt::ptr(&c));
};

for (size_t i = 0; i < 5; ++i) {
  fmt::println("Outer value: {} &outer: {}", c, fmt::ptr(&c));
  func();
  ++c;
}
```

Lambda functions: capture by reference

```cpp
// Capture by reference
int c{42};
auto func = [&c]() {
  fmt::println("Inner value: {} &inner: {}", c, fmt::ptr(&c));
};

for (size_t i = 0; i < 5; ++i) {
  fmt::println("Outer value: {} &outer: {}", c, fmt::ptr(&c));
  func();
  ++c;
}
```

Lambda functions: capture all

```cpp
// Capture everything by value
int c{42};

auto func = [=](){
  fmt::println("Inner value: {}" , c );
};

for(size_t i{} ; i < 5 ;++i){
  fmt::println("Outer value: {}" , c );
  func();
  ++c;
}


// Capturing all reference
int c{ 42 };
int d{ 5 };

auto func = [&]() {
  fmt::println("Inner value: {}", c);
  fmt::println("Inner value(d): {}", d);
};

for (size_t i{}; i < 5; ++i) {
  fmt::println("Outer value: {}", c);
  func();
  ++c;
}
```