Slides


# Section :Control Flow

Slide intentionally left empty

# Flow Control :Introduction

```cpp
bool red = false;
bool green {true};
bool yellow {false};
bool police_stop{true};

/*
 *      If green : go
 *      If red, yellow : stop
 *      If green and police_stop : stop
 * */

 if(red){
     std::cout << "Stop" << std::endl;
 }
 if(yellow){
     std::cout << "Slow down" << std::endl;
 }
 if(green){
     std::cout << "Go" << std::endl;
 }
```

if

else

switch

Ternary operator

Slide intentionally left empty

# If statement

# Doing things conditionally

## if clause

```
int number1 {55};
int number2 {60};

bool result = (number1 < number2);//Expression yielding the condition
std::cout << std::boolalpha << "result : " << result << std::endl;

std::cout << std::endl;
std::cout << "free standing if statement" << std::endl;

//if(result){
if(result == true){
    std::cout << number1 << " is less than " << number2 << std::endl;
}

//if(!result){
if(!(result == true)){
    std::cout << number1 << " is NOT less than " << number2 << std::endl;
}
```

## else clause

```
//Using else
std::cout << std::endl;
std::cout << "using the else clause : " << std::endl;

if(result == true){
    std::cout << number1 << " is less than " << number2 << std::endl;
}else{
    std::cout << number1 << " is NOT less than " << number2 << std::endl;
}
```

## expression as condition

```
//Use expression as condition directly
std::cout << std::endl;
std::cout << "Using expression as condition : " << std::endl;

if(number1 < number2){
    std::cout << number1 << " is less than " << number2 << std::endl;
}else{
    std::cout << number1 << " is NOT less than " << number2 << std::endl;
}
```

```cpp
bool red = false;
bool green {true};
bool yellow {false};
bool police_stop{true};

/*
 *      If green : go
 *      If red, yellow : stop
 *      If green and police_stop : stop
 * */

 if(red){
     std::cout << "Stop" << std::endl;
 }
 if(yellow){
     std::cout << "Slow down" << std::endl;
 }
 if(green){
     std::cout << "Go" << std::endl;
 }
```

## Nested conditions

```
std::cout << std::endl;
std::cout << "Police officer stops(verbose)" << std::endl;
if(green){
    if(police_stop){
        std::cout << "Stop" << std::endl;
    }
    else{
        std::cout << "Go" << std::endl;
    }
}
```

## Nesting alternative

```
std::cout << std::endl;
std::cout << "Police officer stops(less verbose)" << std::endl;
if(green && !police_stop){
    std::cout << "Go" << std::endl;
}else{
    std::cout << "Stop" << std::endl;
}
```

Slide intentionally left empty

# Else If

# Testing for several different conditions

```
// Tools
const int Pen{ 10 };
const int Marker{ 20 };
const int Eraser{ 30 };
const int Rectangle{ 40 };
const int Circle{ 50 };
const int Ellipse{ 60 };
```

# else if clauses

```
int tool{ Ellipse };

if (tool == Pen) {
    std::cout << "Active tool is pen" << std::endl;
    //Do the actual painting
}
else if (tool == Marker) {
    std::cout << "Active tool is Marker" << std::endl;
}
else if (tool == Eraser) {
    std::cout << "Active tool is Eraser" << std::endl;
}
else if (tool == Rectangle) {
    std::cout << "Active tool is Rectangle" << std::endl;
}
else if (tool == Circle) {
    std::cout << "Active tool is Circle" << std::endl;
}
else if (tool == Ellipse) {
    std::cout << "Active tool is Ellipse" << std::endl;
}
```


MainWindow
File Edit Tools Help
Pen Properties
Pen Color
Pen Width :
11
Pen Style :
Solid
Brush Properties
Brush Color :
Brush Style :
Solid
Canvas Properties
Show Grid
Center Scene
Scene Background :
Ellipse
Quick
Rectangle
Star
Hello
World
Learn
QtQuick

Slide intentionally left empty

# Switch

# Testing for several different conditions

```
// Tools
const int Pen{ 10 };
const int Marker{ 20 };
const int Eraser{ 30 };
const int Rectangle{ 40 };
const int Circle{ 50 };
const int Ellipse{ 60 };
```

# switch

```
int tool{ Pen };

switch (tool) {

    case Pen: {
        std::cout << "Active tool is pen" << std::endl;
    }
    break;

    case Marker: {
        std::cout << "Active tool is Marker" << std::endl;
    }
    break;

    default: {
        std::cout << "Can't match any tool" << std::endl;
    }
}
```

Break;

The break statement after each case is very important. It stops processing the switch block when a successful case has been found. If the break statement is not there, all the cases following the current case will be executed.

| | |
|---|---|
| condition | Integral types and enums : int, long, unsigned short, etc. |


MainWindow
File Edit Tools Help
Pen Properties
Pen Color
Pen Width :
11
Pen Style :
Solid
Brush Properties
Brush Color :
Brush Style :
Solid
Canvas Properties
Show Grid
Center Scene
Scene Background :
Ellipse
Quick
Rectangle
Star
Hello
World
Learn
QtQuick

Slide intentionally left empty

# Short Circuit Evaluations

AND

If one of the operands is “false”, the result is false. Put operands likely to be false first.

OR

If one of the operands is “true”, the result is true. Put operands likely to be true first.

```cpp
bool a {true};
bool b {true};
bool c {true};
bool d {false};

bool p{false};
bool q{false};
bool r{false};
bool m{true};

//AND : If one of the operands is 0, the result is 0
std::cout << std::endl;
std::cout << "AND short circuit" << std::endl;
bool result = a && b && c && d;
std::cout << "result : " << std::boolalpha << result << std::endl;

//OR : If one of the operands is 1, the result is 1.
std::cout << std::endl;
std::cout << "OR short circuit" << std::endl;
result = p || q || r || m;
std::cout << "result : " << std::boolalpha << result << std::endl;
```

Why care ?

Computing some of the operands in the expression can be expensive. If short circuit is possible, such expensive computations can be avoided.

## Test conditions wrapped into functions

```
bool car() {
    std::cout << "car function running" << std::endl;
    return true;
}

bool house() {
    std::cout << "house function running" << std::endl;
    return false;
}
bool job() {
    std::cout << "job function running" << std::endl;
    return true;
}
bool spouse() {
    std::cout << "spouse function running" << std::endl;
    return true;
}
```

## Short circuit evaluation in action

```
if (car() && house() && job() && spouse()) {
    std::cout << "I am happy" << std::endl;
}
else {
    std::cout << "I am sad" << std::endl;
}
```

Slide intentionally left empty

# Integral Logic Conditions

| | |
|---|---|
| Any number different than 0, or expression evaluating to something other than 0 | true |

Any number equal to 0, or expression evaluating to 0

false

```cpp
int item_count{-3};

if(item_count){
    std::cout << "We have items in the bag. " << item_count << " to be exact" << std::endl;
}else{
    std::cout << "Sorry ! You have no item in the bag" << std::endl;
}
```

Slide intentionally left empty

# Ternary Operator

```cpp
int max{};

int a{35};
int b{20};

std::cout << std::endl;
std::cout << "using regular if " << std::endl;

if(a  >  b){
    max = a;
}else{
    max = b;
}

std::cout << "max : " << max << std::endl;
```

## Ternary expression

```
result = (condition) ? option1 : option2 ;
```

## Equivalent

```
if(condition){
    result = option1;
}else{
    result = option2;
}
```

## Ternary

```
int max{};

int a{35};
int b{20};

max = (a > b) ? a : b ;

std::cout << "max : " << max << std::endl;
```

Types must much or be convertible

```
max = (a > b) ? a : "b" ;// Error
```

## Ternary initialization

```
//Ternary Initialization
std::cout << std::endl;
std::cout << "speed" << std::endl;
bool fast = false;

int speed { fast ? 300 : 150};

std::cout << "The speed is : " << speed << std::endl;
```

Slide intentionally left empty

# if constexpr

IDE

Compile Time

Run Time

```cpp
#include <iostream>

int main(int argc, char **argv)
{
    std::cout << "Hello World in C++20!" << std::endl;
    return 0;
}
```

Compiler

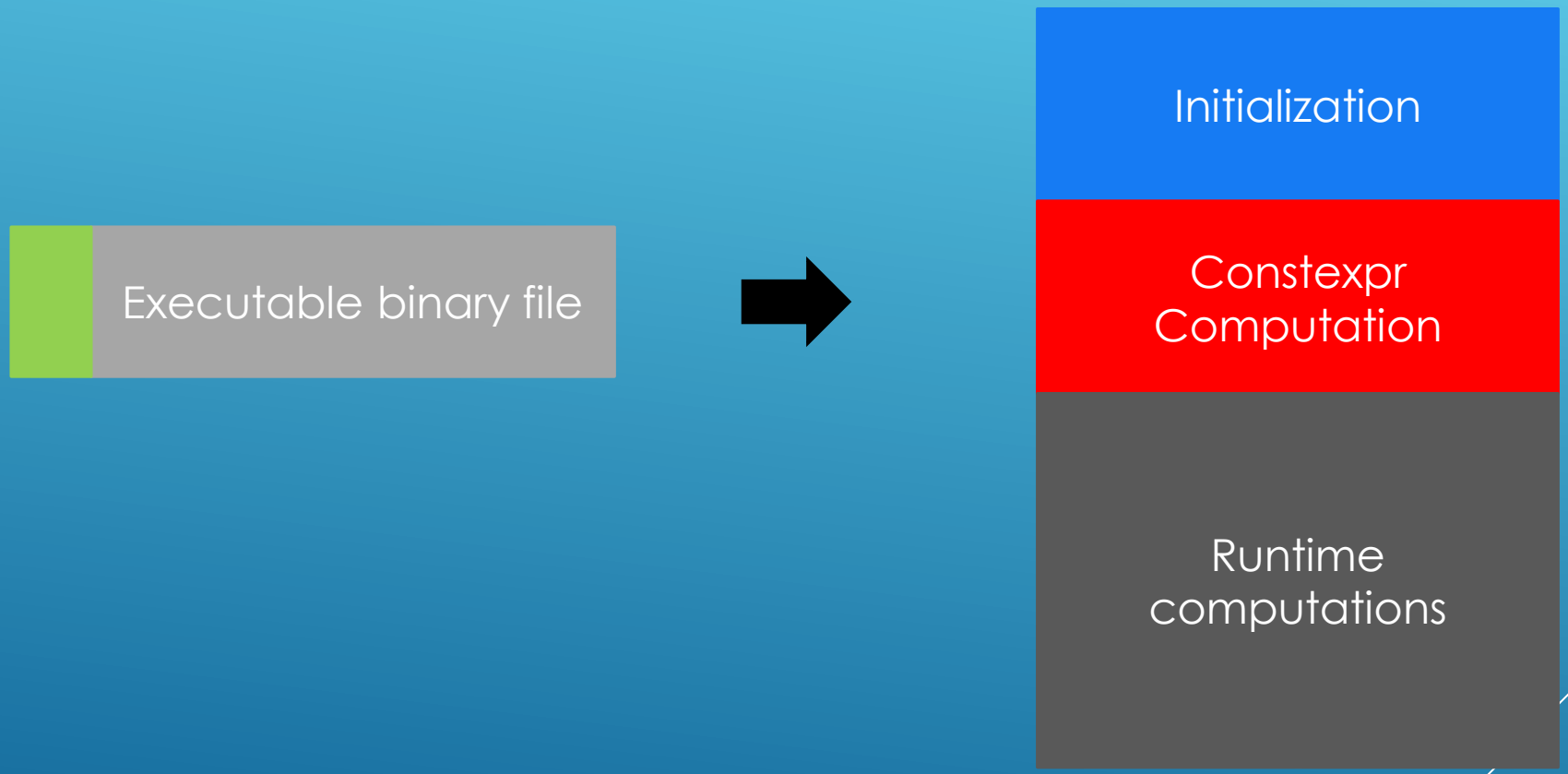
Executable binary file

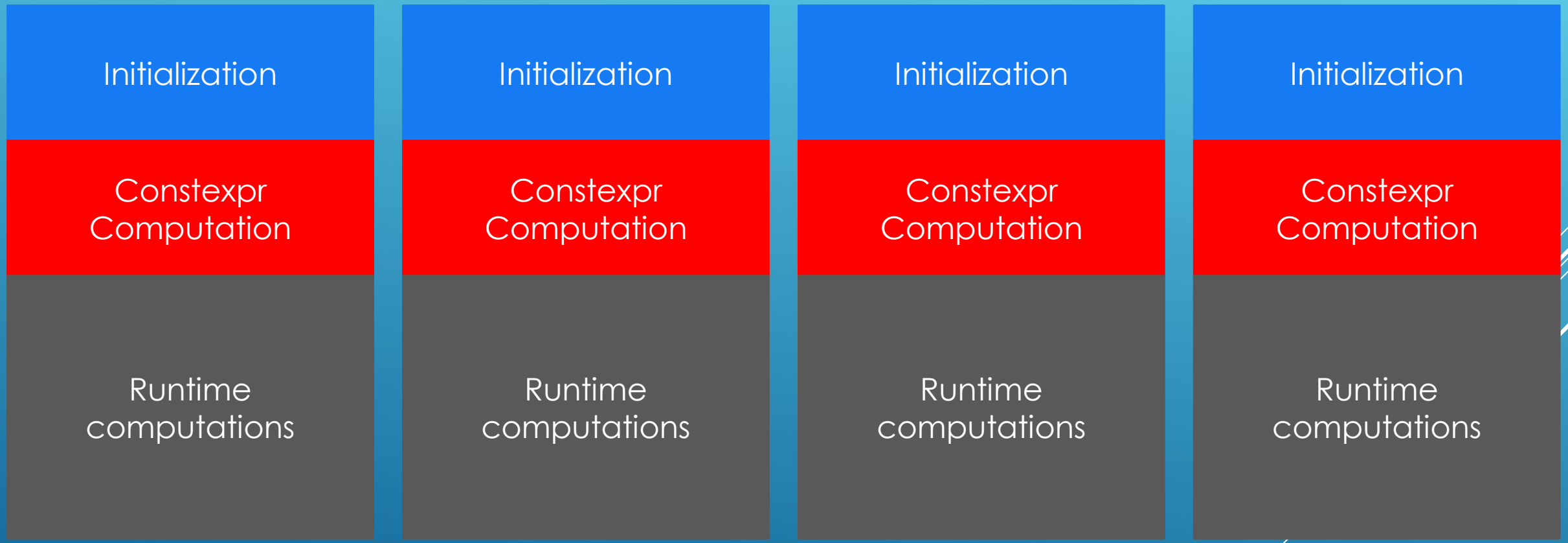
```cpp
#include <iostream>

int main(int argc, char **argv)
{
    std::cout << "Hello World in C++20!" << std::endl;
    return 0;
}
```

Compiler

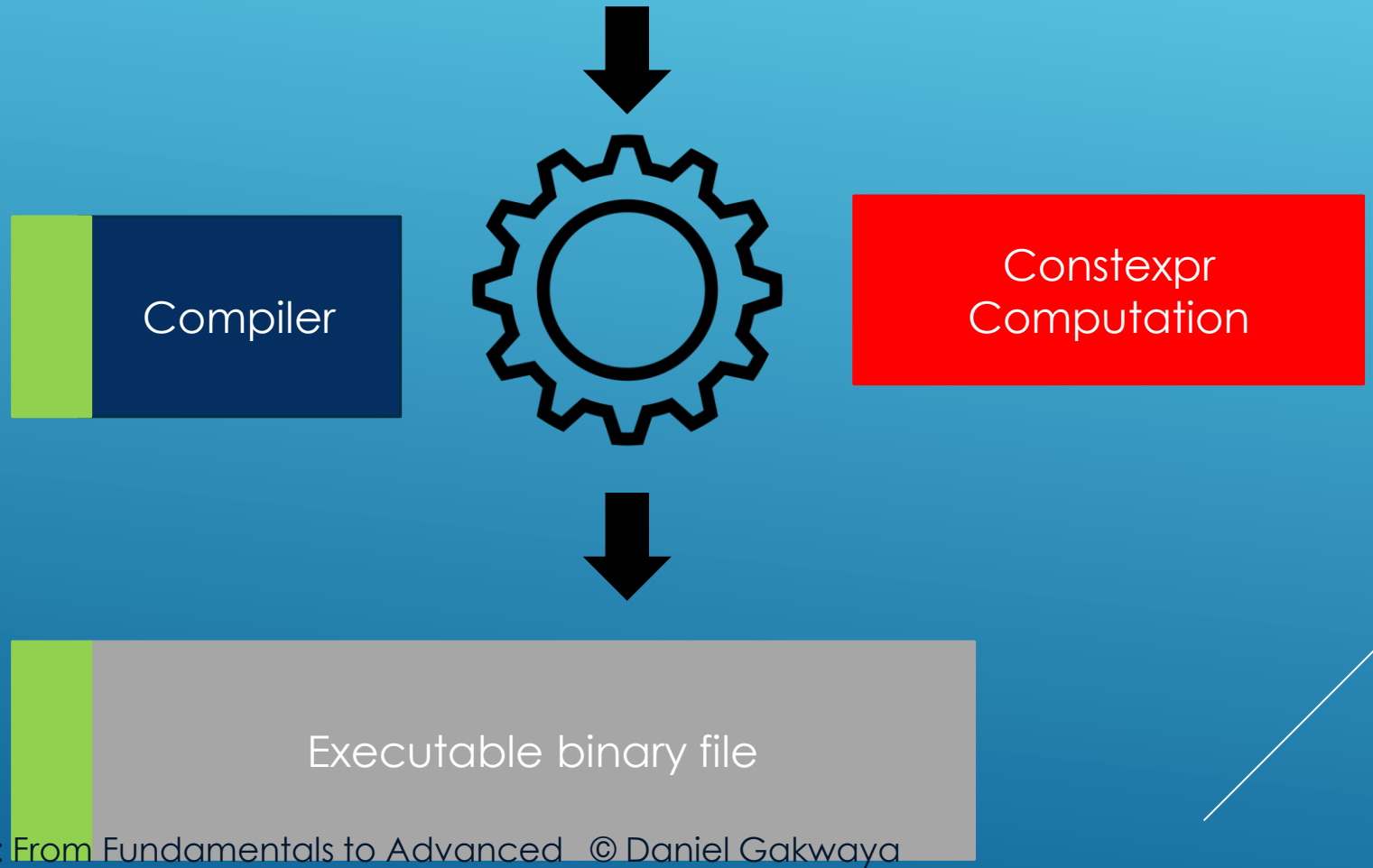
Executable binary file

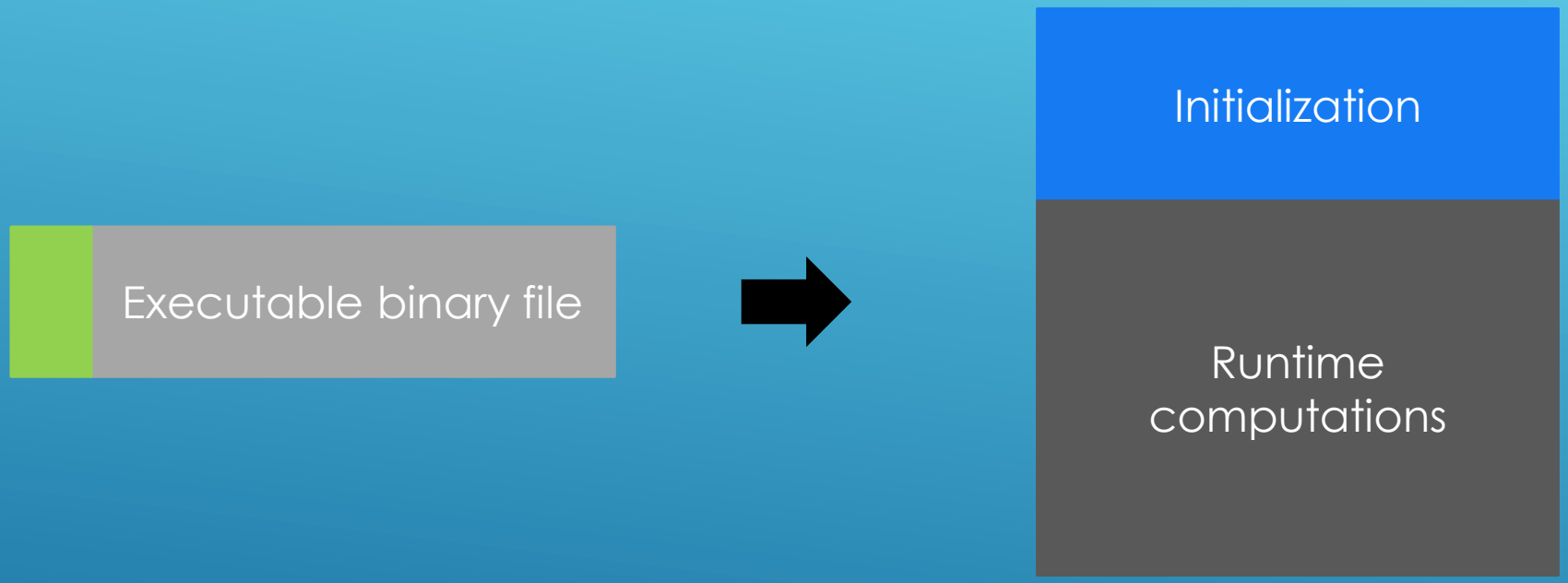

Executable binary file
Initialization
Constexpr Computation
Runtime computations

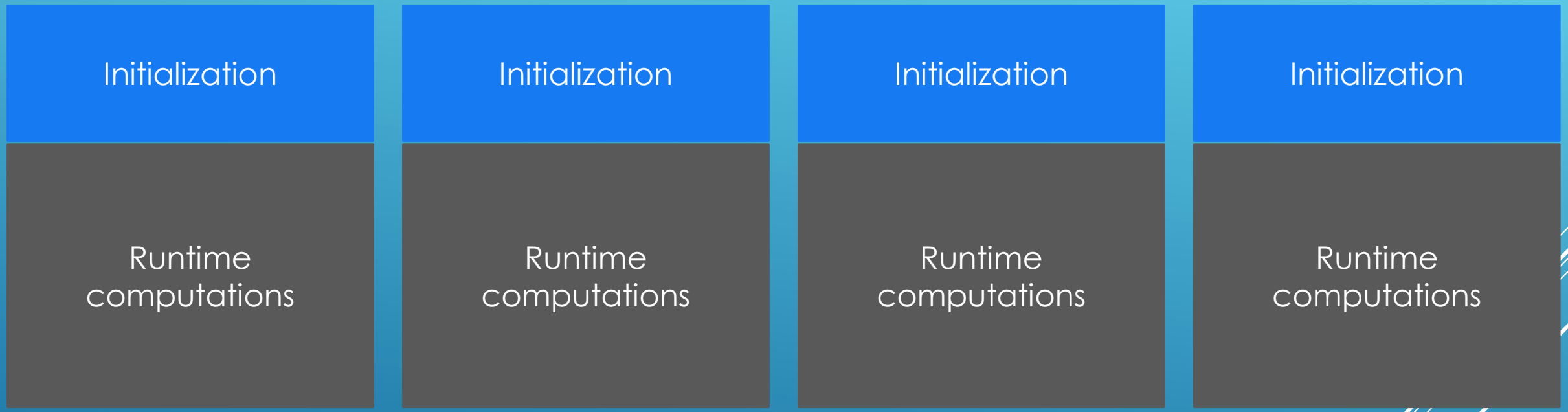

Initialization
Constexpr Computation
Runtime computations
Initialization
Constexpr Computation
Runtime computations
Initialization
Constexpr Computation
Runtime computations
Initialization
Constexpr Computation
Runtime computations

```cpp
#include <iostream>

int main(int argc, char **argv)
{
    std::cout << "Hello World in C++20!" << std::endl;
    return 0;
}
```




Executable binary file
Initialization
Runtime computations


Initialization
Runtime computations
Initialization
Runtime computations
Initialization
Runtime computations
Initialization
Runtime computations

Runtime computations


Compile time computations
Runtime computations


Compile time computations
Runtime computations


Compile time computations
Runtime computations


Compile time computations
Runtime computations


Compile time computations
Runtime computations

```
constexpr bool condition{false};

if constexpr(condition){
    std::cout << "Condition is true" << std::endl;
}else{
    std::cout << "Condition is not true" << std::endl;
}
```

Slide intentionally left empty

# if initializer

```cpp
int speed {10};

bool go {false};

if(go){

    if(speed > 5){
        std::cout << "Slow down!" << std::endl;
    }else{
        std::cout << "All good!" << std::endl;
    }
}else{
    std::cout << "Stop" << std::endl;
}

std::cout << "speed : " << speed << std::endl;
```

```
//With if initializer
if( int high_speed{33};go){

    if(high_speed > 5){
        std::cout << "Slow down!" << std::endl;
    }else{
        std::cout << "All good!" << std::endl;
    }

}else{
    std::cout << "high_speed : " << high_speed << std::endl;
    std::cout << "Stop" << std::endl;
}
```

Slide intentionally left empty

# switch initializer

```
int tool {Eraser};

switch (double strength{3.56};tool)
{
    case Pen : {
        std::cout << "Active tool is Pen. strength : " << strength <<  std::endl;
    }
    break;

    case Marker : {
        std::cout << "Active tool is Marker. strength : " << strength <<  std::endl;
    }
    break;

    default: {
        std::cout << "No match found. strength : " << strength <<  std::endl;
    }
        break;
}
```

Slide intentionally left empty

# Scope with if statements

# Scope

```
#include <iostream>

int global_var{45};

int main(int argc, char **argv)
{
    bool green{false};

    if(green){
        int car_count{23};
        std::cout << "The color is green" << car_count << " cars on the move!" << std::endl;
    }else{
        std::cout << "The color is not green. Y'all should stop!" << std::endl;
    }

    return 0;
}
```

Slide intentionally left empty

# Switch Scope

## No {} on cases

```
int int_condition{ 0 };

switch (int_condition) {
case 0:
    std::cout << "We are dealing with 0" << std::endl;
    std::cout << "Another statement in case 0" << std::endl;
    break;
case 1:
    std::cout << "We are dealing with 1" << std::endl;
    std:: cout << "Some other statement" << std::endl
    break;
default:
    std::cout << "We are not dealing with 0 or 1" << std::endl;
    std::cout << "Some other statement" << std::endl;
    break;// Optional but good practice
}
```

# Giant scope

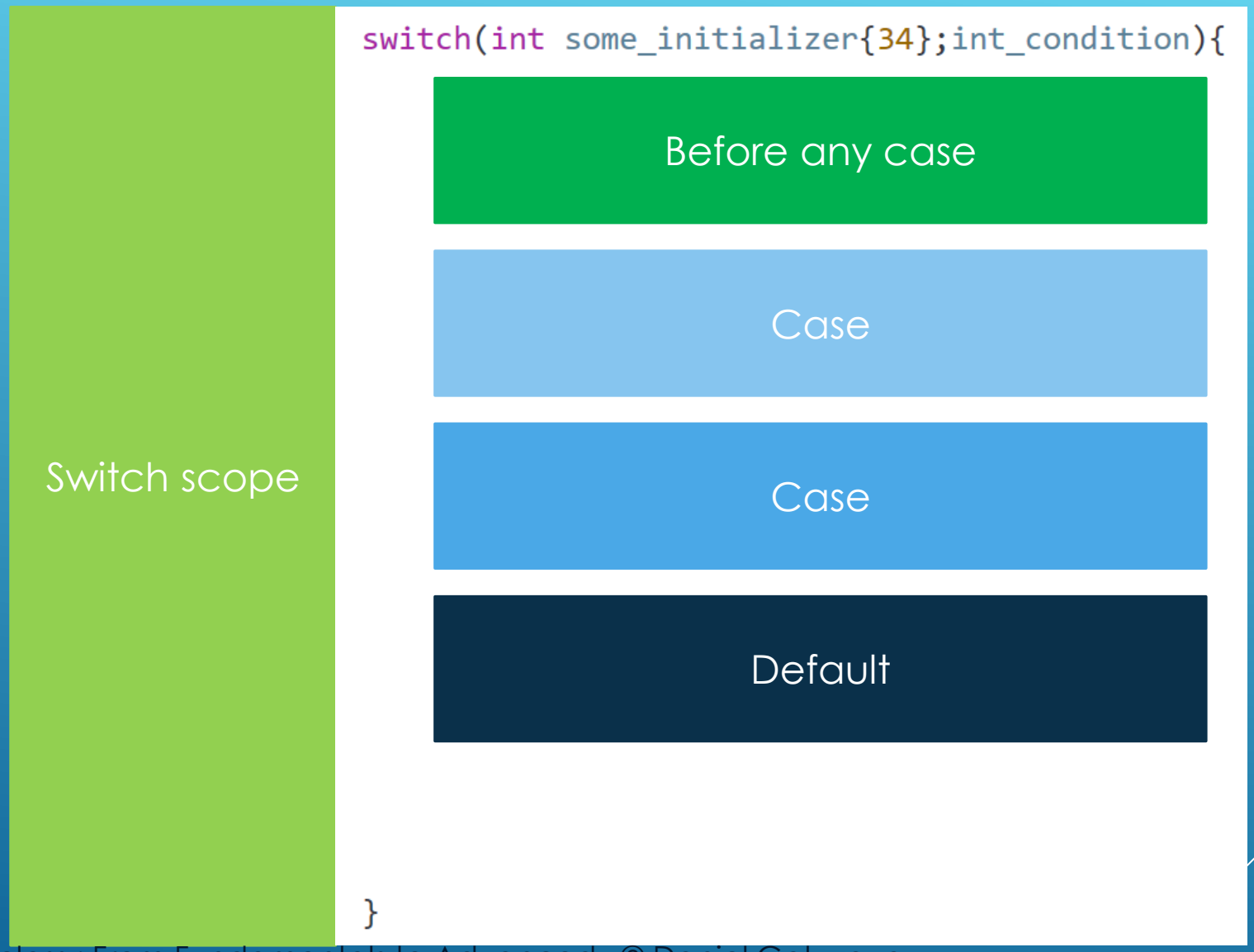
```
int int_condition{ 0 };

switch (int_condition) {
    int x;
case 0:
    int y;
    x = 4;
    x++;
    break;
case 1:
    int z;
    y = 5;
    y += 5;
    break;
default:
    int u;
    z = 4;
    z++;
    break;// Optional but good practice
}
```

# Initialization in cases

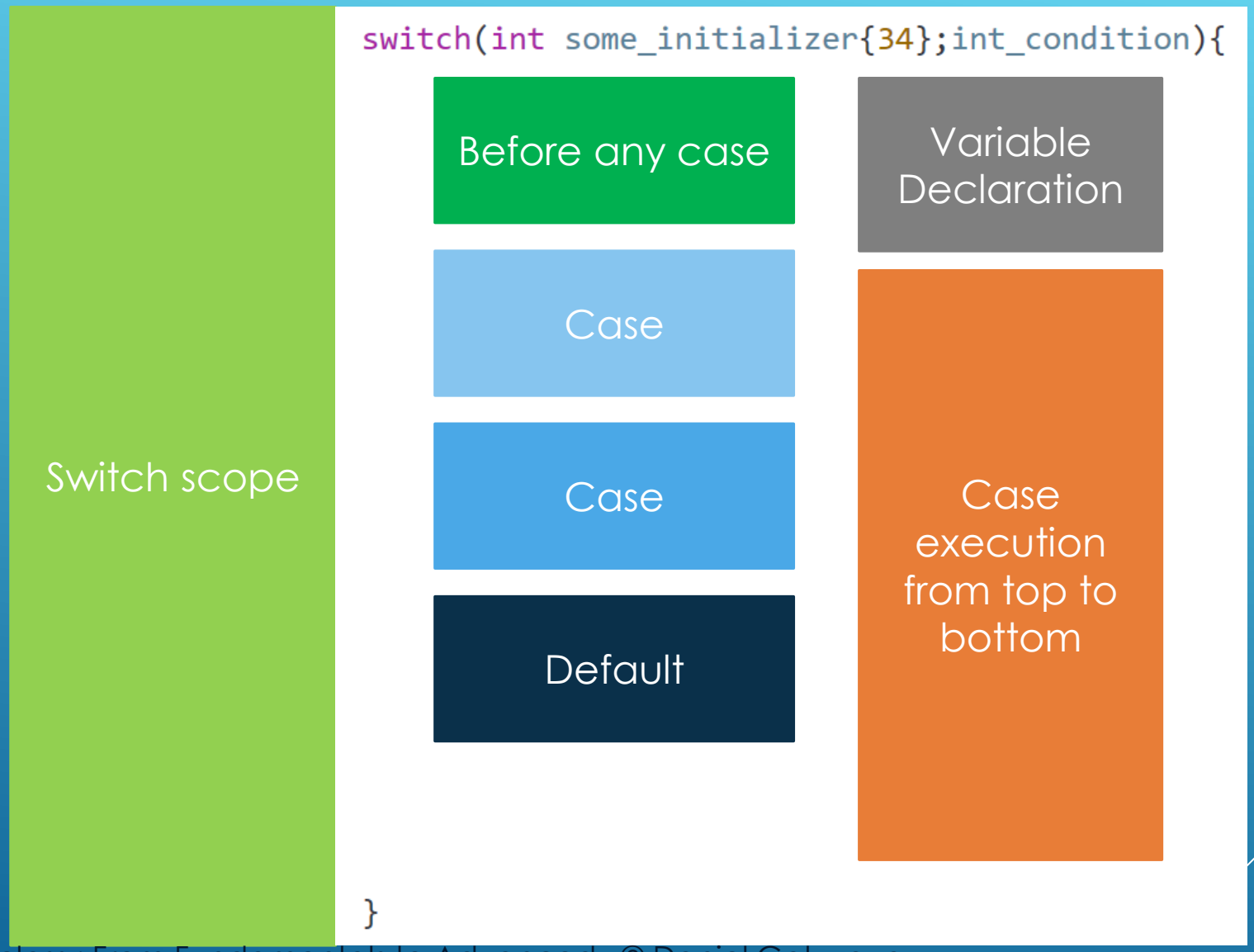
```
int int_condition{ 0 };

switch (int_condition) {
    int x{5}; // Compiler error
case 0:
    int y{4}; // Compiler error
    x = 4;
    x++;
    break;
case 1:
    int z{6}; // Compiler error
    y = 5;
    y += 5;
    break;
default:
    int u{8}; // OK
    z = 4;
    z++;
    break;// Optional but good practice
}
```


Switch scope
switch(int some_initializer{34};int_condition){
Before any case
Case
Case
Default
}

Switch scope

Everything in a switch block lives in a single giant scope


Switch scope
switch(int some_initializer{34};int_condition){
Before any case
Variable Declaration
Case
Case execution from top to bottom
Case
Default
}

Warning

Your can't refer to a variable before the case under which it is declared in a switch block

Recommendation

While the compiler allows all the weird things we just experienced, I strongly recommend not declaring variables inside the switch block. Use an initializer of a plain old variable outside the switch block instead.

Slide intentionally left empty

# Flow Control :Summary

```cpp
bool red = false;
bool green {true};
bool yellow {false};
bool police_stop{true};

/*
 *      If green : go
 *      If red, yellow : stop
 *      If green and police_stop : stop
 * */

 if(red){
     std::cout << "Stop" << std::endl;
 }
 if(yellow){
     std::cout << "Slow down" << std::endl;
 }
 if(green){
     std::cout << "Go" << std::endl;
 }
```

if

else

switch

Ternary operator