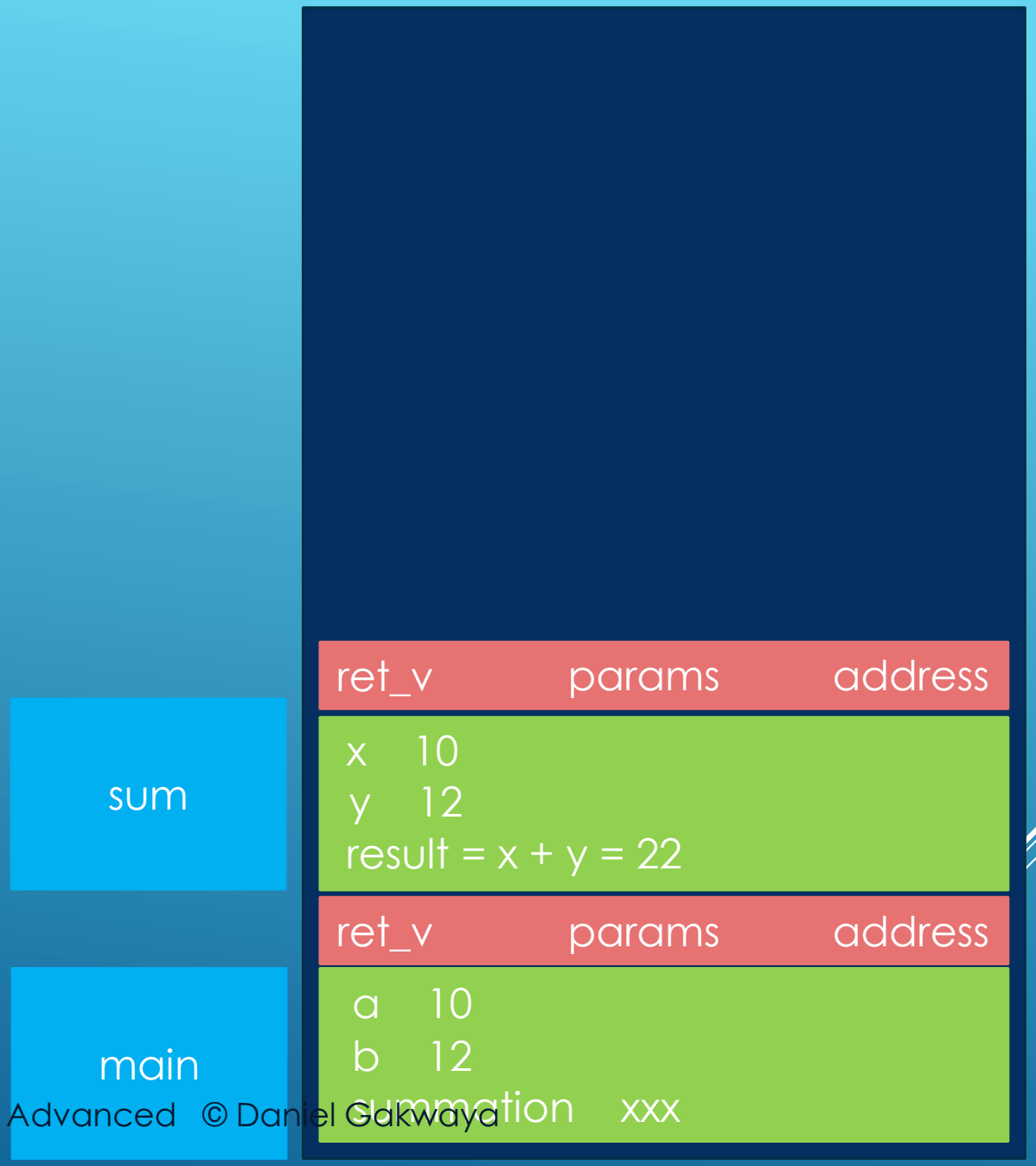
Slides

Section :Function call stack and debugging

Slide intentionally left empty

Function call stack & Debugging

```cpp
int main(int argc, char **argv){

    int a{10};
    int b{12};

    int summation  =  sum( a , b);
    std::cout << "sum : "
        << summation << std::endl;

    return 0;
}

int& adjust( int& input){
    int adjustment{2};
    input += adjustment;
    return input;
}

int sum(int x , int y){

    int result = x + y;
    adjust(result);
    return result;
}
```



- Visual Studio Code
- CodeLite
- Microsoft Visual Studio


Locals
Watches
Call Stack

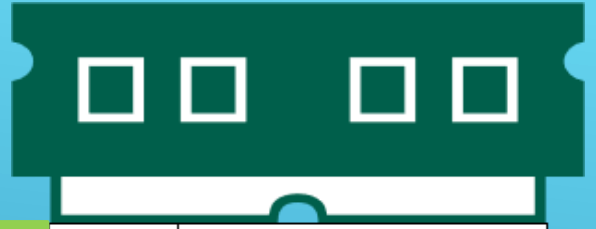
Slide intentionally left empty

Function call stack

```cpp
#include <iostream>

int add_numbers(int a, int b)
{
    return a + b;
}

int main()
{
    int a = 10;
    int b = 5;
    int c;

    std::cout << "Statement1" << std::endl;
    std::cout << "Statement2" << std::endl;
    c = add_numbers(a, b);
    std::cout << "Statement3" << std::endl;
    std::cout << "Statement4" << std::endl;

    return 0;
}
```

Compiler

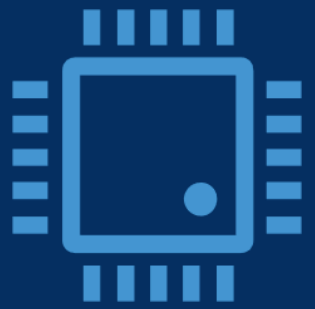
```
a = 10      (int)
b = 5       (int)
c           (int)
print("Statement1")
print("Statement2")
c = f_add(a,b)
print("Statement3")
print("Statement4")
end
```



Program area

0001	
0002	
0003	
0004	
0005	
0006	
0007	
0008	
0009	
0010	
...	
...	
0020	
...	
...	
0030	
...	
...	

CPU



0001	
0002	
0003	
0004	
0005	
0006	
0007	

Hard Drive

```
a = 10      (int)
b = 5       (int)
c           (int)
print("Statement1")
print("Statement2")
c = f_add(a,b)
print("Statement3")
print("Statement4")
end
```

Heap

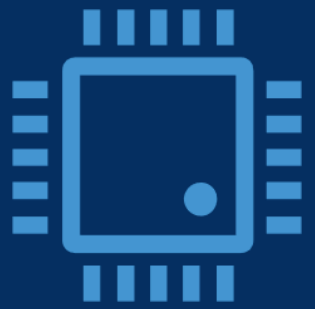
Stack

static and global variables

code Section

0001	
0002	
0003	
0004	
0005	
0006	
0007	
0008	
0009	
0010	
...	
...	
0020	
...	
...	
0030	
...	
...	

CPU

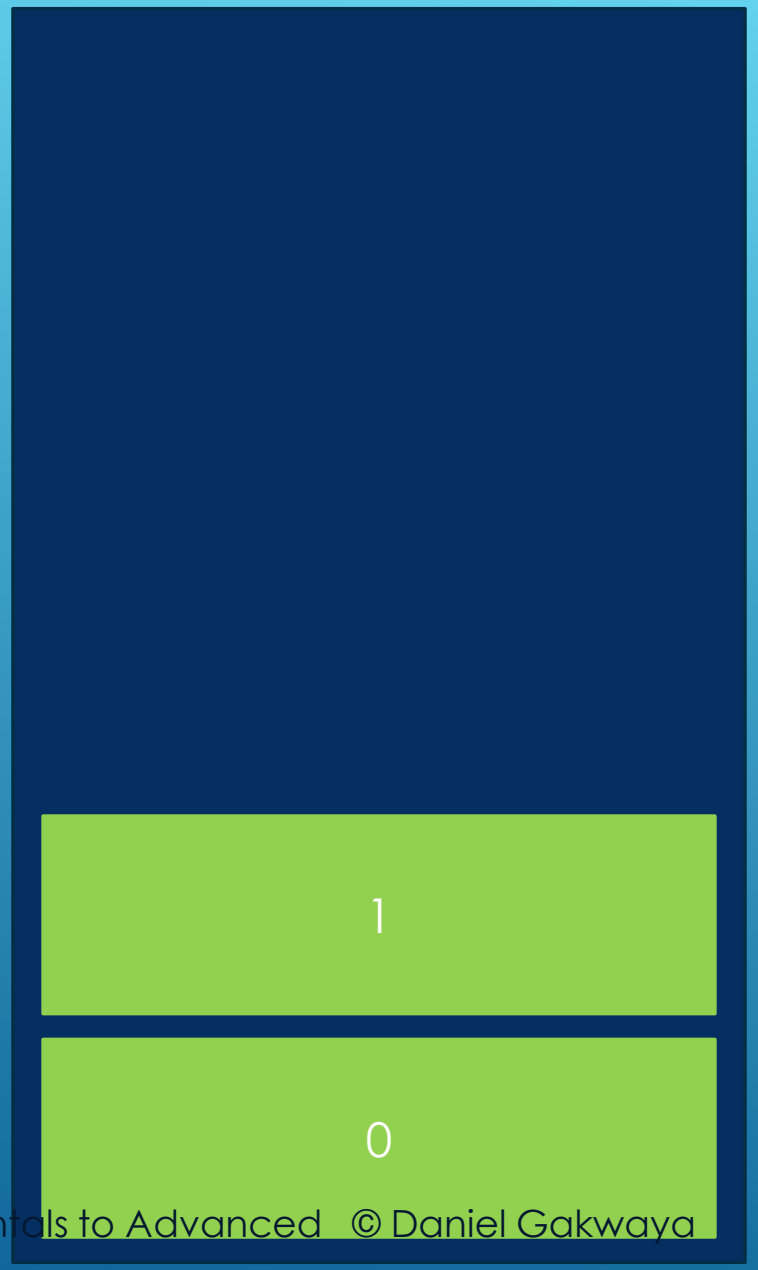


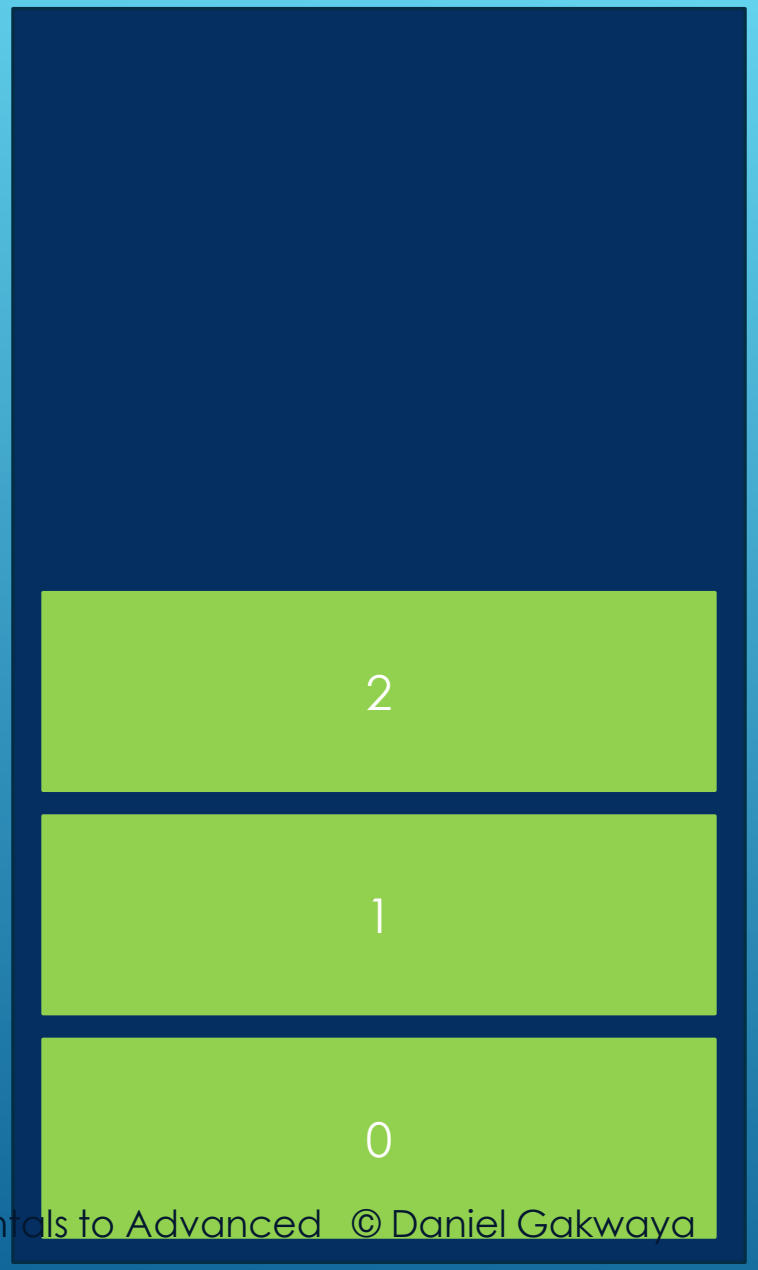
0001	
0002	
0003	
0004	
0005	
0006	
0007	

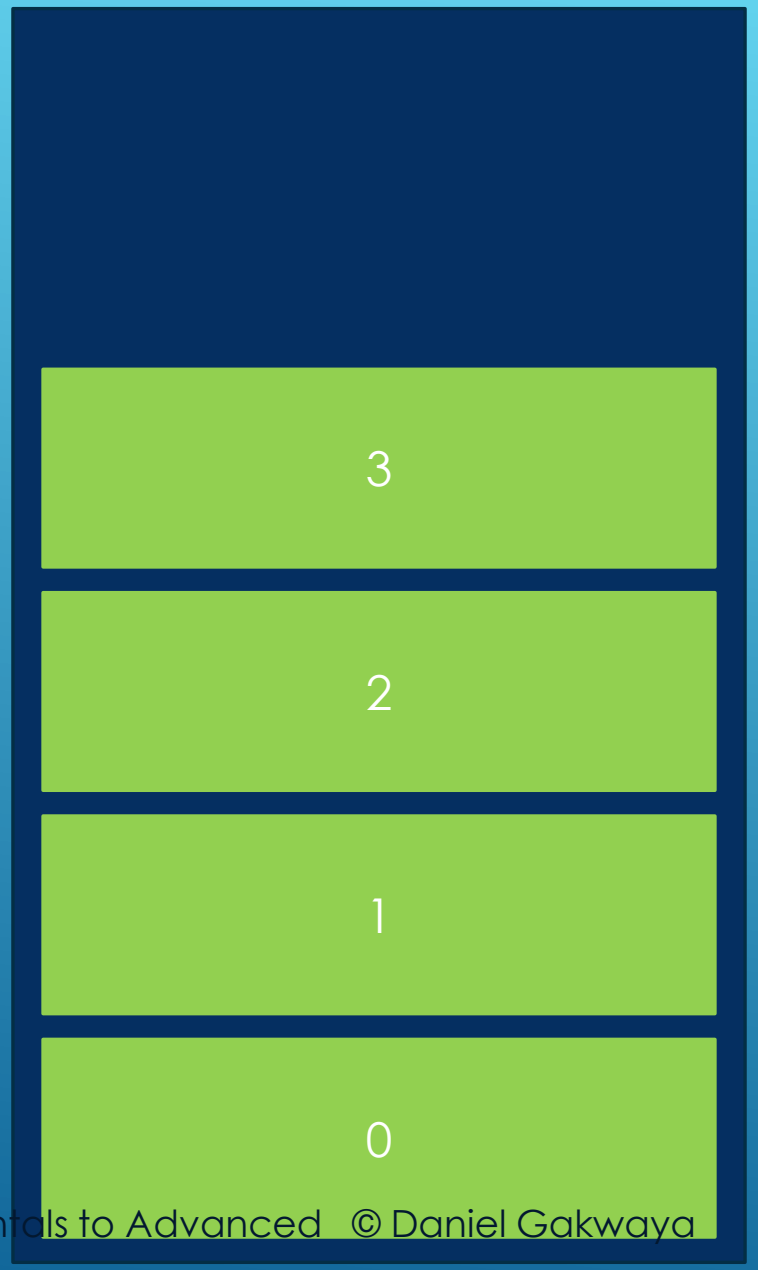
Hard Drive

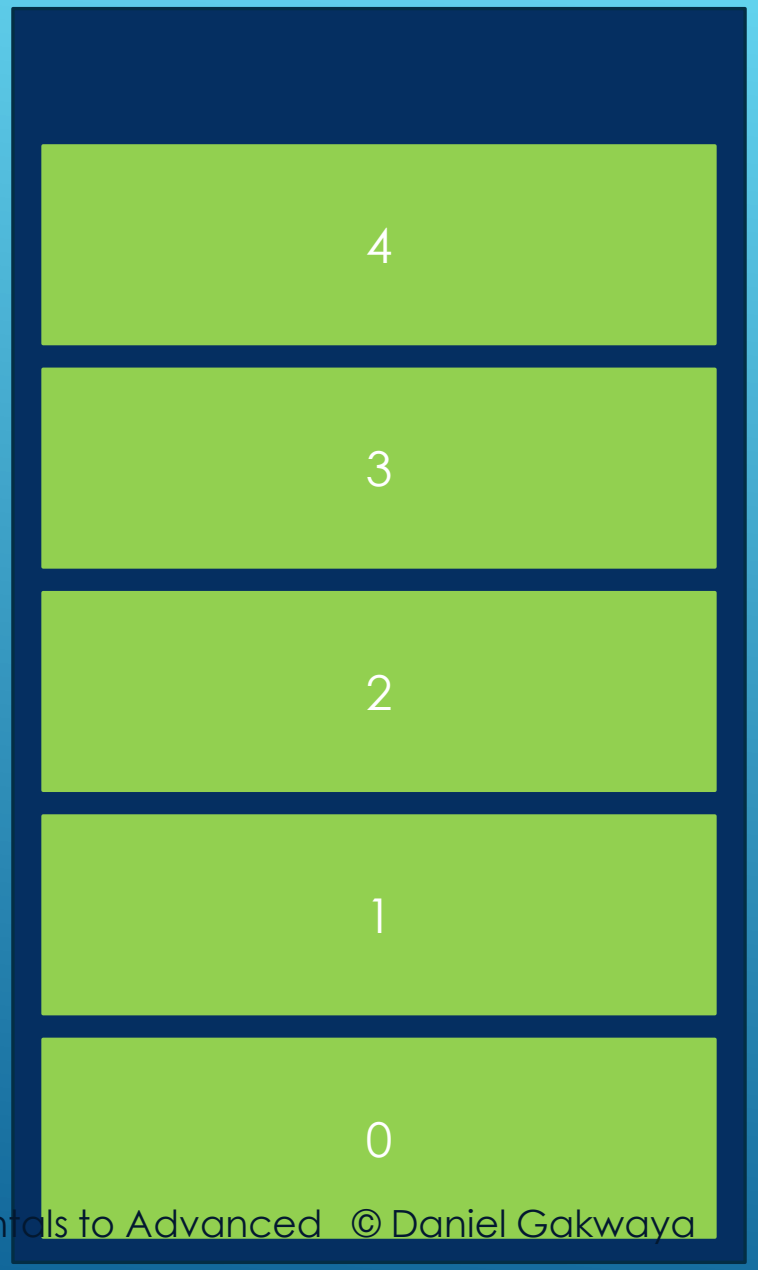
```
a = 10        (int)
b = 5         (int)
c             (int)
print("Statement1")
print("Statement2")
c = f_add(a,b)
print("Statement3")
print("Statement4")
end
```

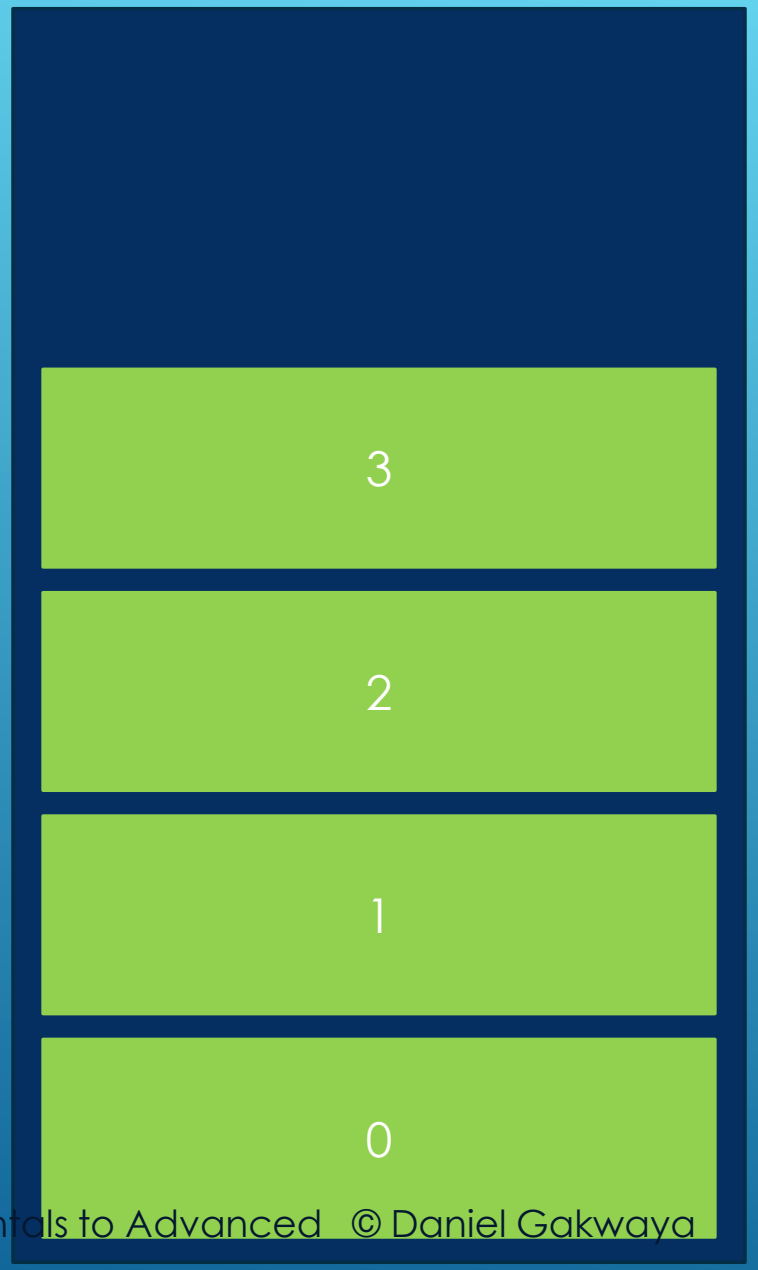

0

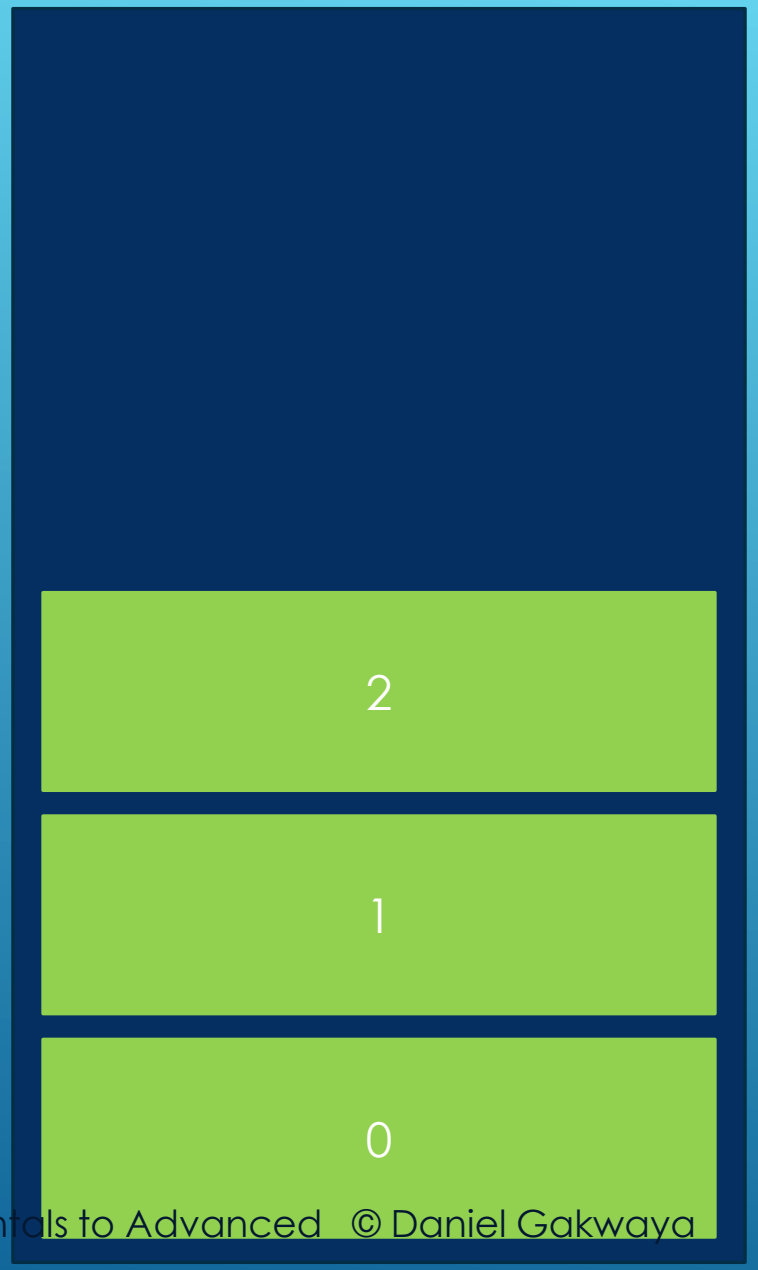

1
0

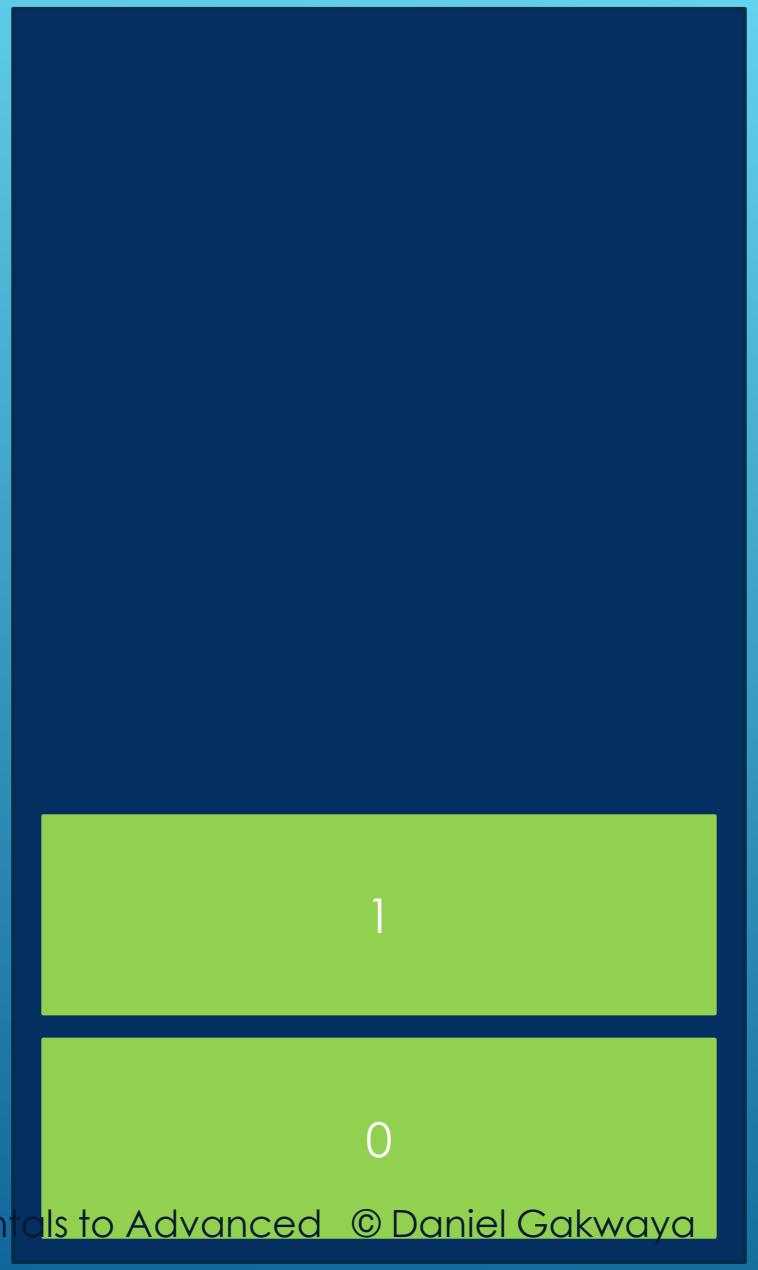

2
1
0


3
2
1
0


4
3
2
1
0


3
2
1
0


2
1
0


1
0


0


LIFO
Last In First Out

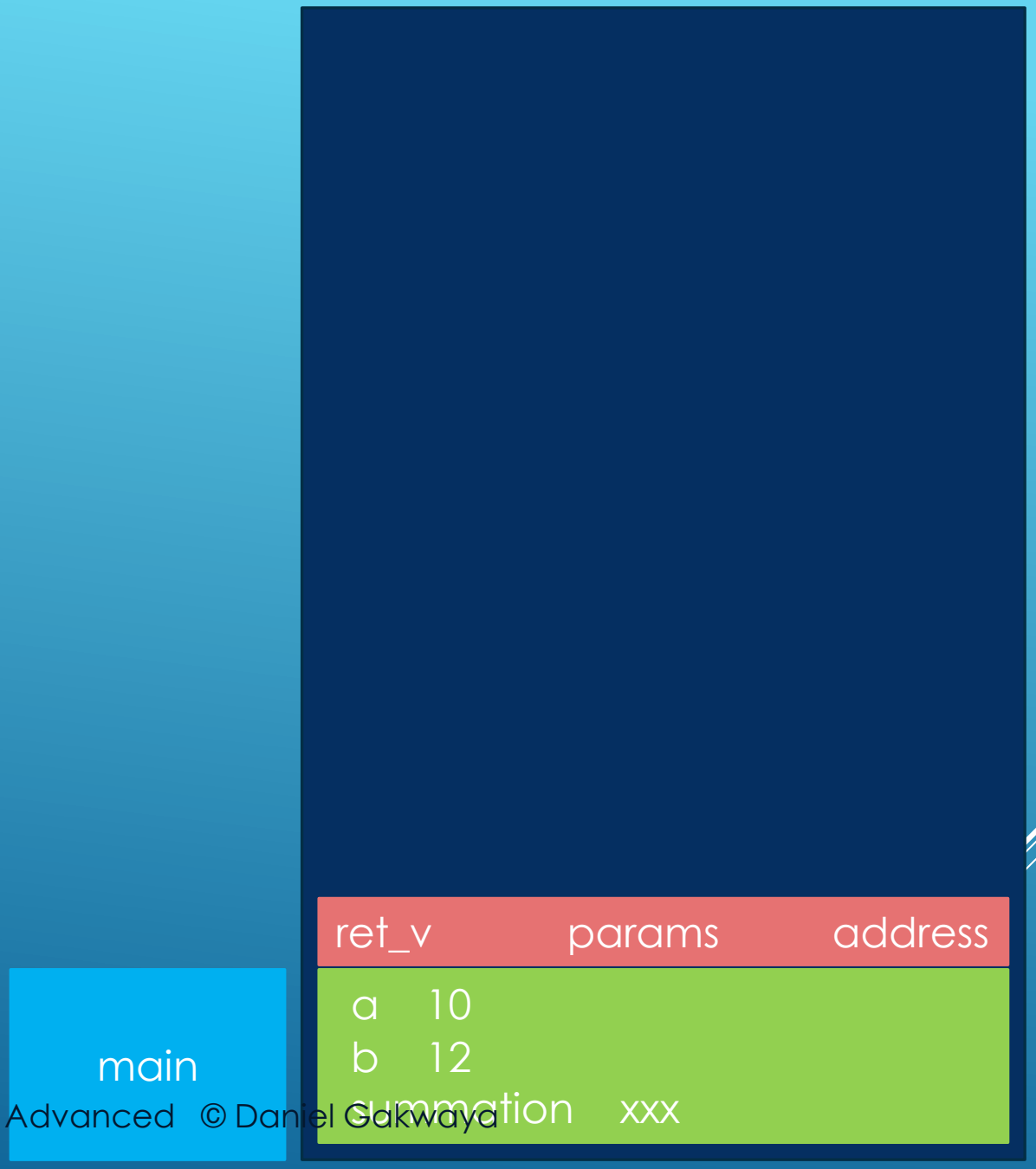
```cpp
int main(int argc, char **argv){

    int a{10};
    int b{12};

    int summation  =  sum( a , b);
    std::cout << "sum : "
        << summation << std::endl;

    return 0;
}

int& adjust( int& input){
    int adjustment{2};
    input += adjustment;
    return input;
}

int sum(int x , int y){

    int result = x + y;
    adjust(result);
    return result;
}
```

```cpp
int main(int argc, char **argv){

    int a{10};
    int b{12};

    int summation  =  sum( a , b);
    std::cout << "sum : "
        << summation << std::endl;

    return 0;
}

int& adjust( int& input){
    int adjustment{2};
    input += adjustment;
    return input;
}

int sum(int x , int y){

    int result = x + y;
    adjust(result);
    return result;
}
```

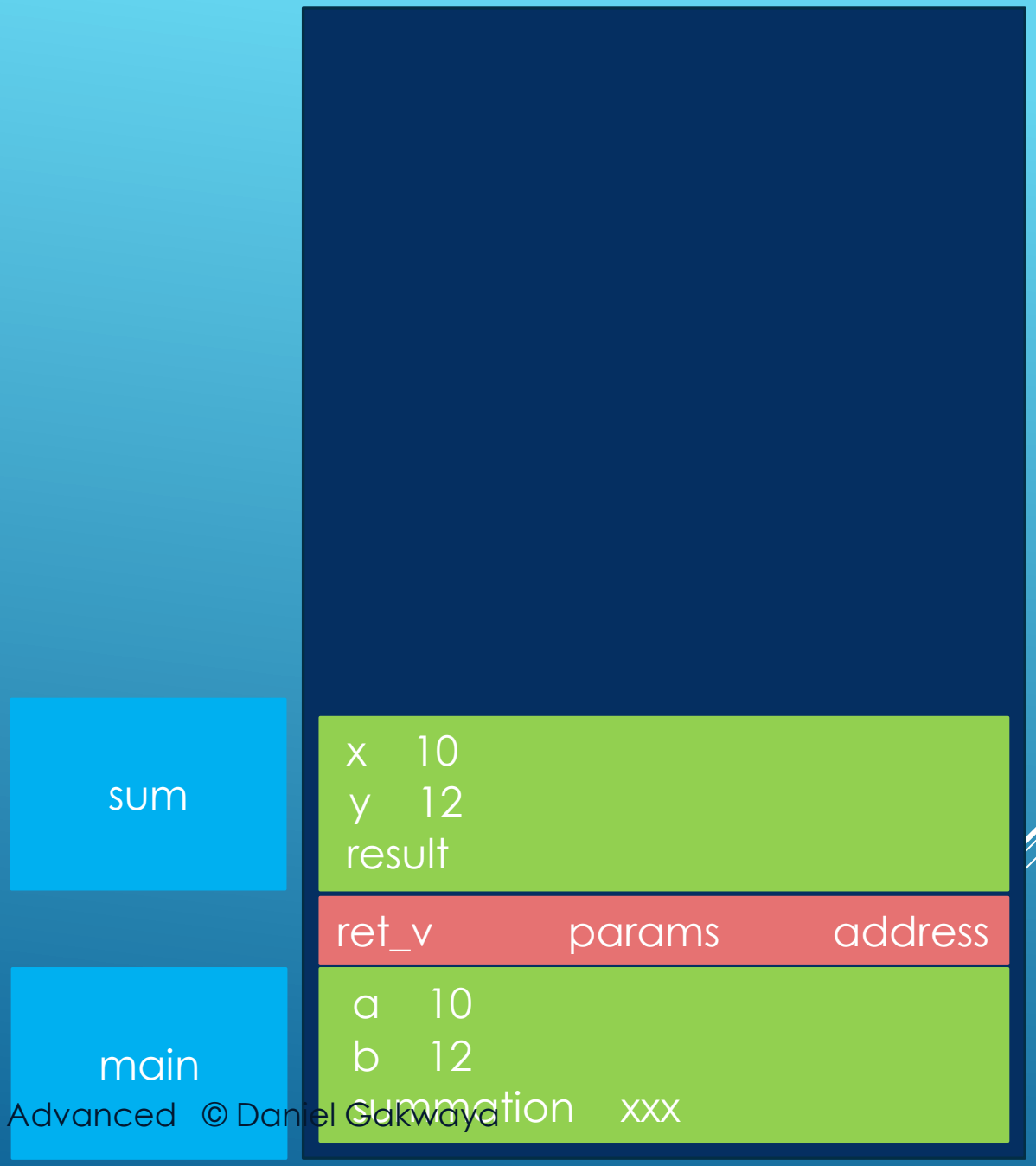
main

a 10
b 12
summation xxx

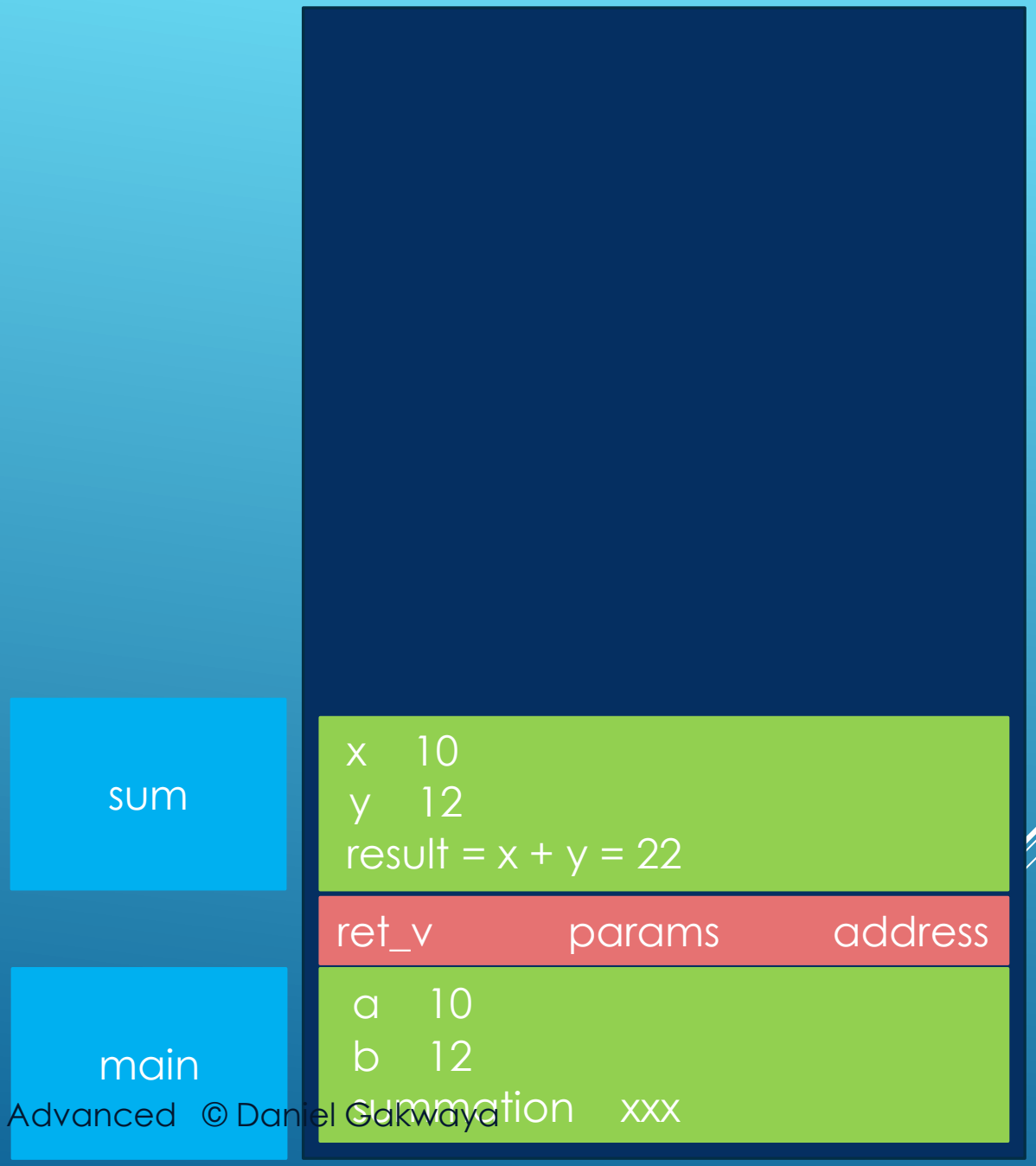
```cpp
int main(int argc, char **argv){

    int a{10};
    int b{12};

    int summation  =  sum( a , b);
    std::cout << "sum : "
        << summation << std::endl;

    return 0;
}

int& adjust( int& input){
    int adjustment{2};
    input += adjustment;
    return input;
}

int sum(int x , int y){

    int result = x + y;
    adjust(result);
    return result;
}
```



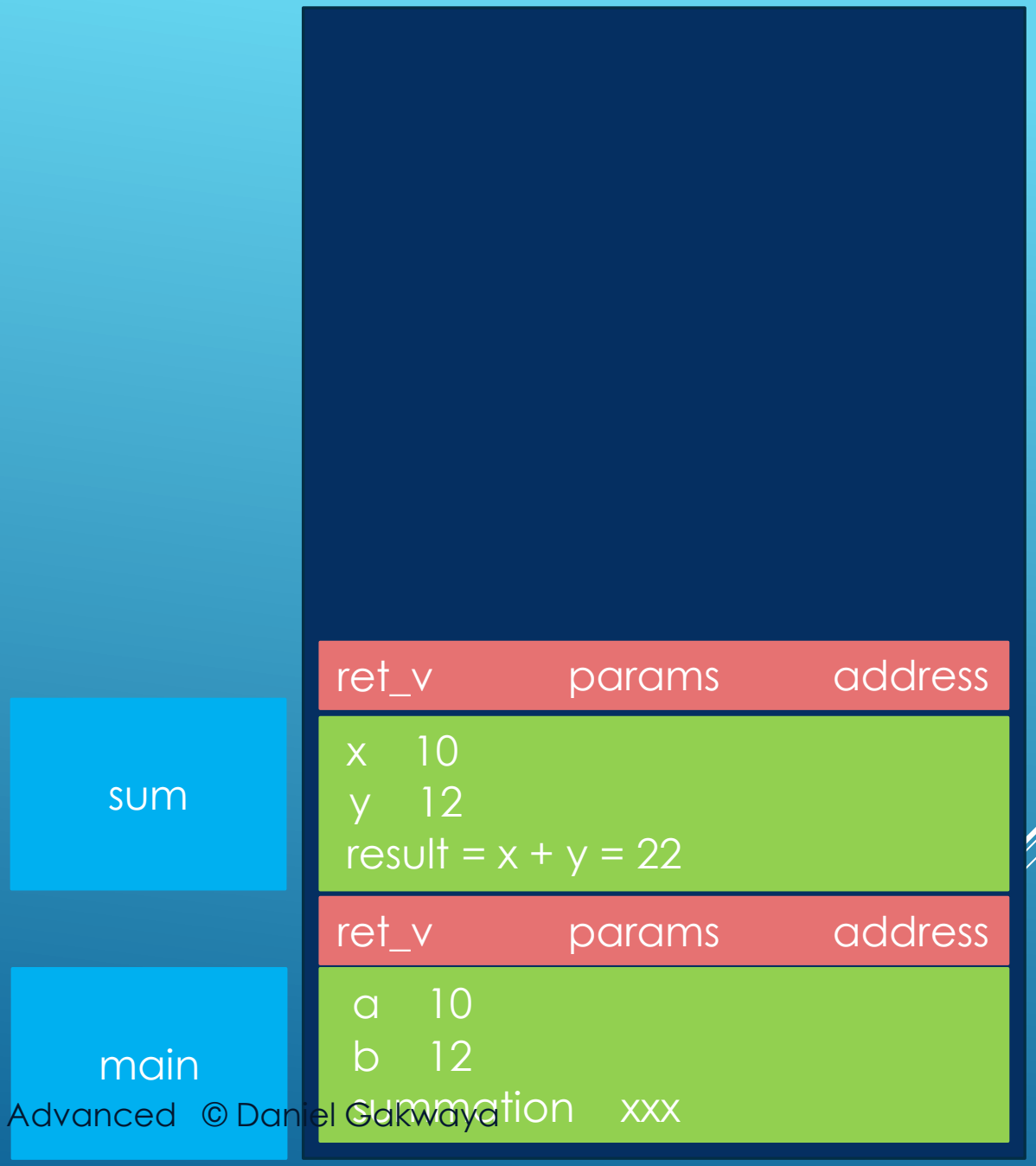
```cpp
int main(int argc, char **argv){

    int a{10};
    int b{12};

    int summation  =  sum( a , b);
    std::cout << "sum : "
        << summation << std::endl;

    return 0;
}

int& adjust( int& input){
    int adjustment{2};
    input += adjustment;
    return input;
}

int sum(int x , int y){

    int result = x + y;
    adjust(result);
    return result;
}
```



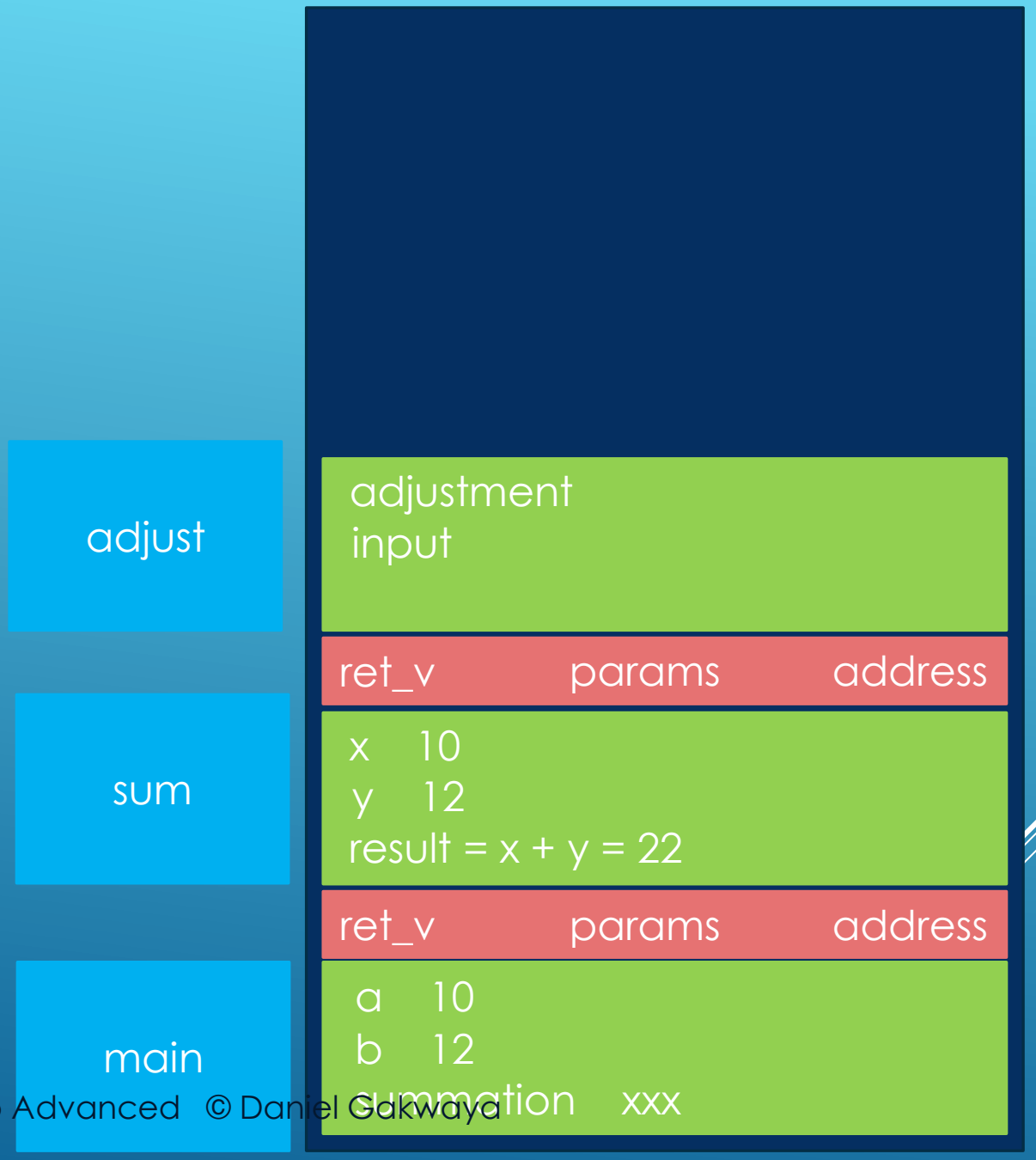
```cpp
int main(int argc, char **argv){

    int a{10};
    int b{12};

    int summation  =  sum( a , b);
    std::cout << "sum : "
        << summation << std::endl;

    return 0;
}

int& adjust( int& input){
    int adjustment{2};
    input += adjustment;
    return input;
}

int sum(int x , int y){

    int result = x + y;
    adjust(result);
    return result;
}
```



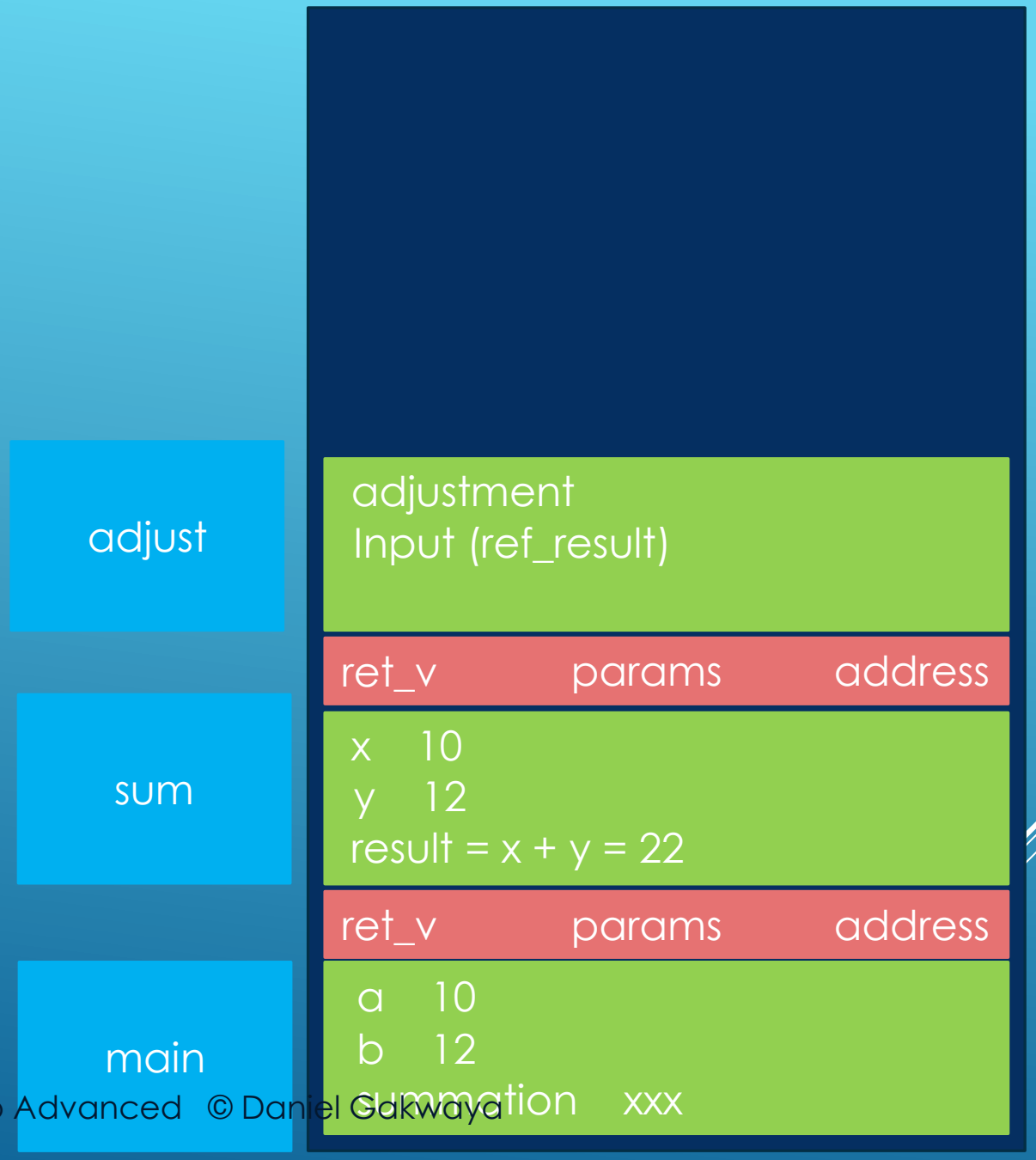
```cpp
int main(int argc, char **argv){

    int a{10};
    int b{12};

    int summation  =  sum( a , b);
    std::cout << "sum : "
        << summation << std::endl;

    return 0;
}

int& adjust( int& input){
    int adjustment{2};
    input += adjustment;
    return input;
}

int sum(int x , int y){

    int result = x + y;
    adjust(result);
    return result;
}
```



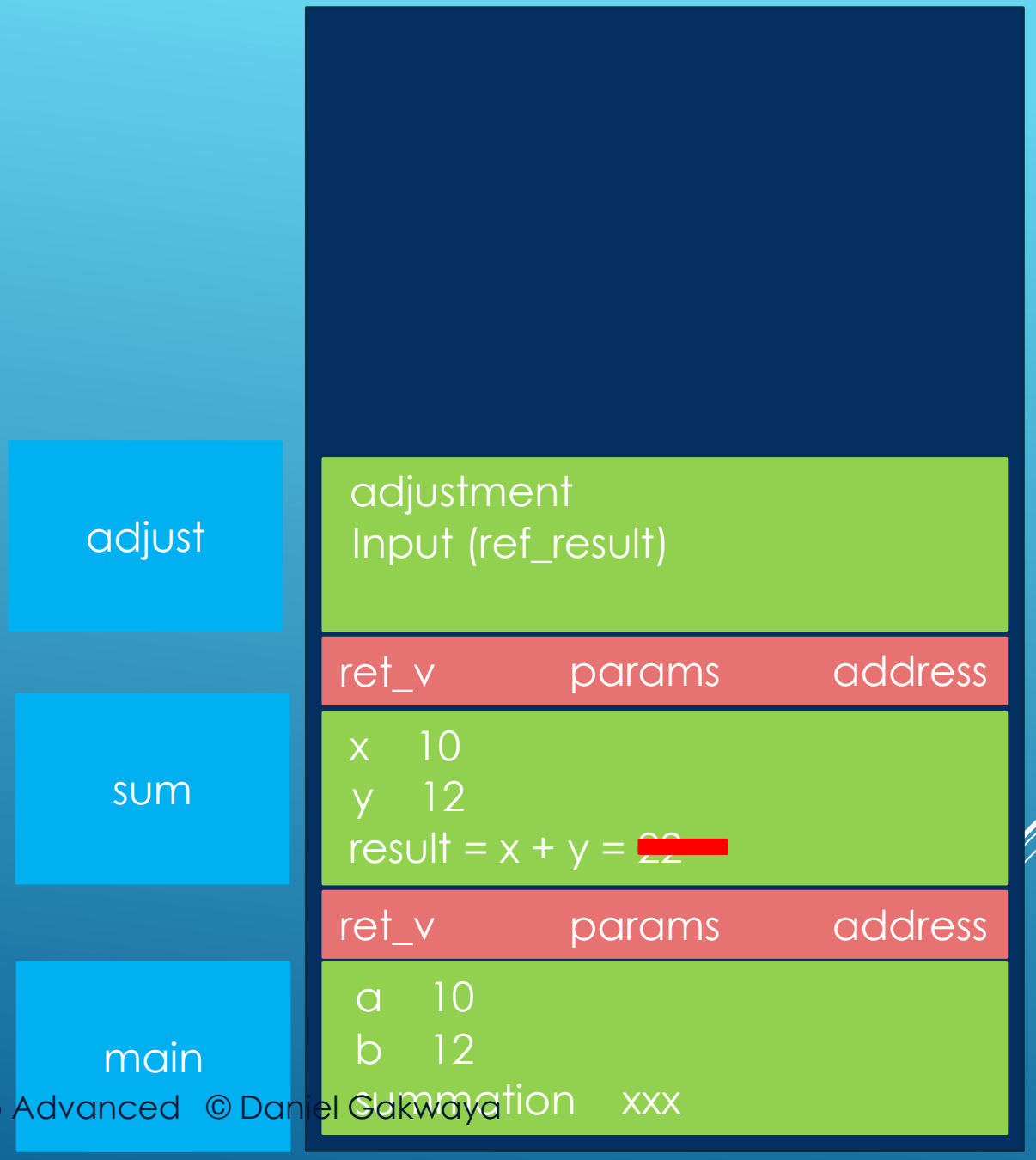
```cpp
int main(int argc, char **argv){

    int a{10};
    int b{12};

    int summation  =  sum( a , b);
    std::cout << "sum : "
        << summation << std::endl;

    return 0;
}

int& adjust( int& input){
    int adjustment{2};
    input += adjustment;
    return input;
}

int sum(int x , int y){

    int result = x + y;
    adjust(result);
    return result;
}
```



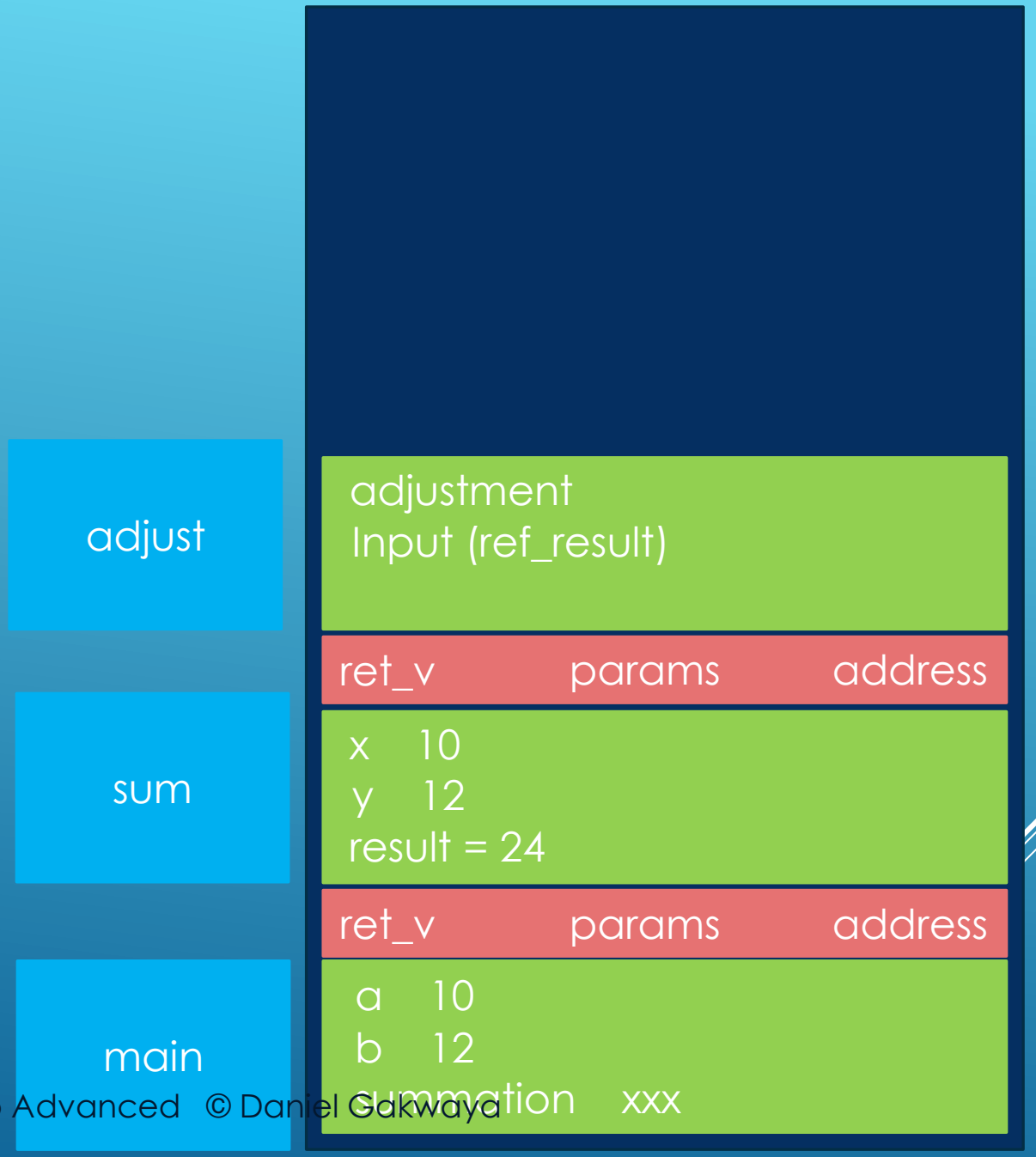
```cpp
int main(int argc, char **argv){

    int a{10};
    int b{12};

    int summation  =  sum( a , b);
    std::cout << "sum : "
        << summation << std::endl;

    return 0;
}

int& adjust( int& input){
    int adjustment{2};
    input += adjustment;
    return input;
}

int sum(int x , int y){

    int result = x + y;
    adjust(result);
    return result;
}
```



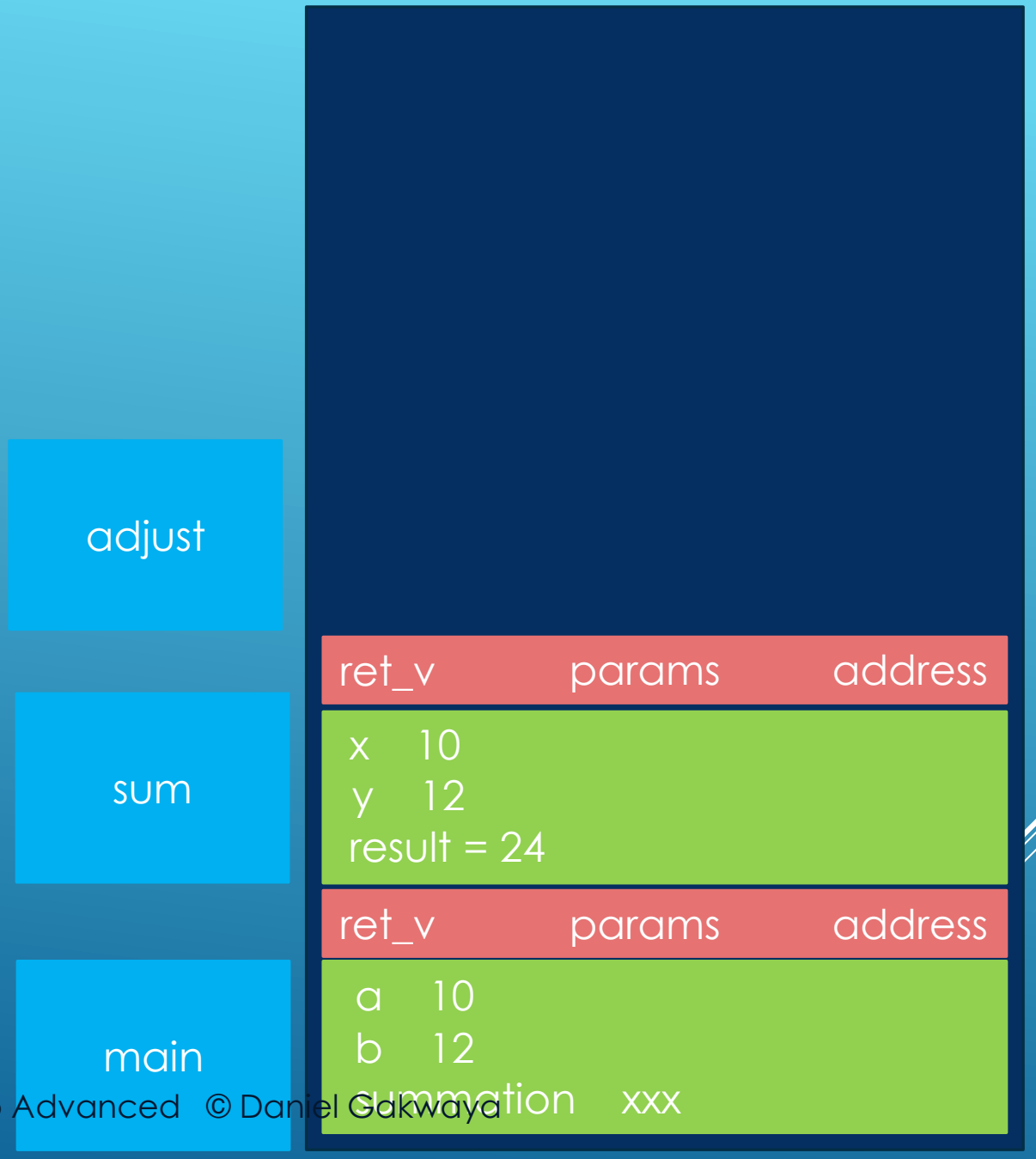
```cpp
int main(int argc, char **argv){

    int a{10};
    int b{12};

    int summation  =  sum( a , b);
    std::cout << "sum : "
        << summation << std::endl;

    return 0;
}

int& adjust( int& input){
    int adjustment{2};
    input += adjustment;
    return input;
}

int sum(int x , int y){

    int result = x + y;
    adjust(result);
    return result;
}
```



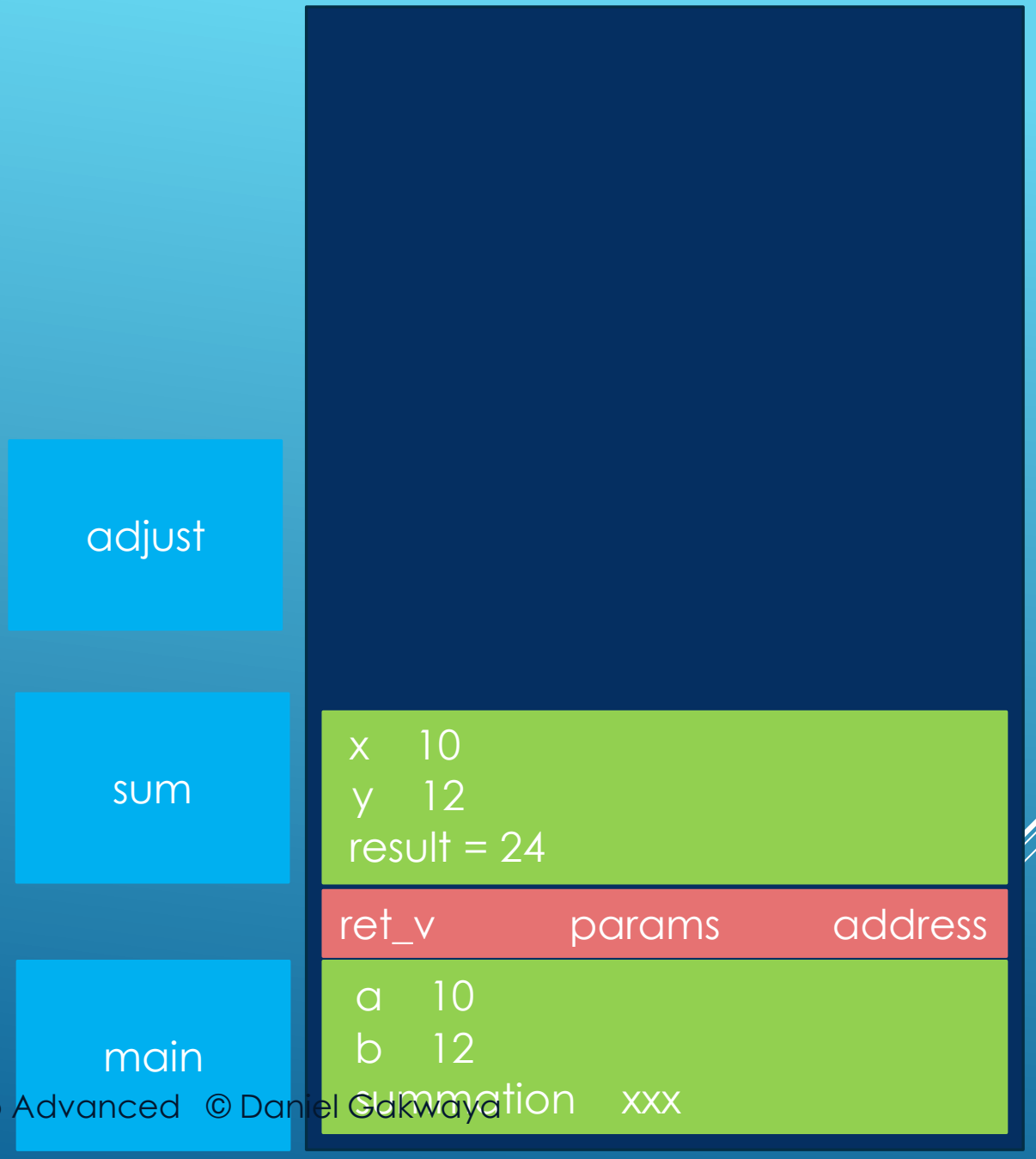
```cpp
int main(int argc, char **argv){

    int a{10};
    int b{12};

    int summation  =  sum( a , b);
    std::cout << "sum : "
        << summation << std::endl;

    return 0;
}

int& adjust( int& input){
    int adjustment{2};
    input += adjustment;
    return input;
}

int sum(int x , int y){

    int result = x + y;
    adjust(result);
    return result;
}
```



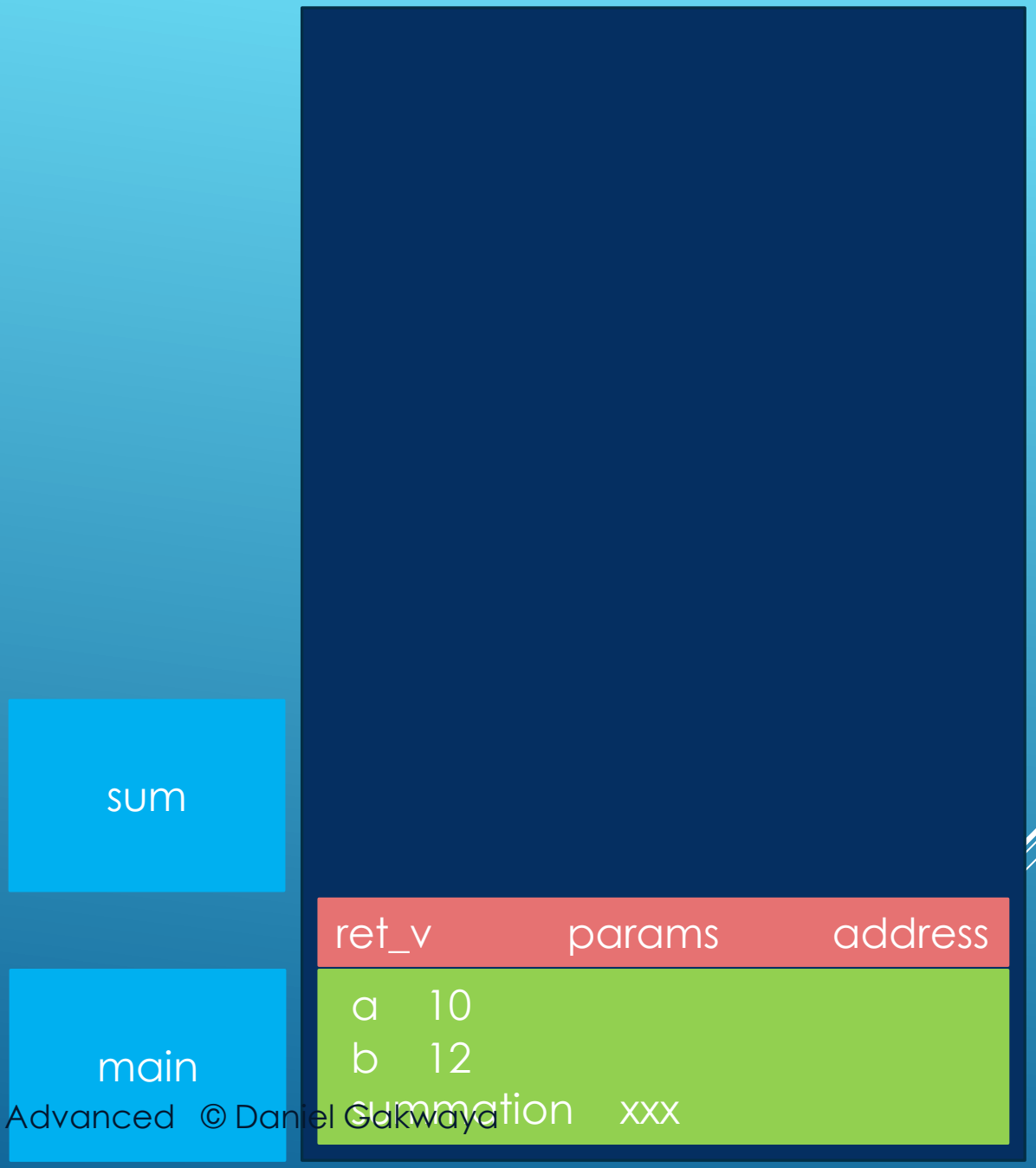
```cpp
int main(int argc, char **argv){

    int a{10};
    int b{12};

    int summation  =  sum( a , b);
    std::cout << "sum : "
        << summation << std::endl;

    return 0;
}

int& adjust( int& input){
    int adjustment{2};
    input += adjustment;
    return input;
}

int sum(int x , int y){

    int result = x + y;
    adjust(result);
    return result;
}
```



```cpp
int main(int argc, char **argv){

    int a{10};
    int b{12};

    int summation  =  sum( a , b);
    std::cout << "sum : "
        << summation << std::endl;

    return 0;
}

int& adjust( int& input){
    int adjustment{2};
    input += adjustment;
    return input;
}

int sum(int x , int y){

    int result = x + y;
    adjust(result);
    return result;
}
```



```cpp
int main(int argc, char **argv){

    int a{10};
    int b{12};

    int summation  =  sum( a , b);
    std::cout << "sum : "
        << summation << std::endl;

    return 0;
}

int& adjust( int& input){
    int adjustment{2};
    input += adjustment;
    return input;
}

int sum(int x , int y){

    int result = x + y;
    adjust(result);
    return result;
}
```



```cpp
int main(int argc, char **argv){

    int a{10};
    int b{12};

    int summation  =  sum( a , b);
    std::cout << "sum : "
        << summation << std::endl;

    return 0;
}

int& adjust( int& input){
    int adjustment{2};
    input += adjustment;
    return input;
}

int sum(int x , int y){

    int result = x + y;
    adjust(result);
    return result;
}
```

main

a 10
b 12
summation xxx

```
int main(int argc, char **argv){

    int a{10};
    int b{12};

    int summation  =  sum( a , b);
    std::cout << "sum : "
        << summation << std::endl;

    return 0;
}

int& adjust( int& input){
    int adjustment{2};
    input += adjustment;
    return input;
}

int sum(int x , int y){

    int result = x + y;
    adjust(result);
    return result;
}
```

main

a 10
b 12
summation 24

Understanding the call stack : some benefits

- Truly understanding how function local variables become alive and die
- The need for inline functions becomes obvious
- Understanding the debug process of your program becomes easier. We are going to learn more about that in the next lecture.

Slide intentionally left empty

Debugging

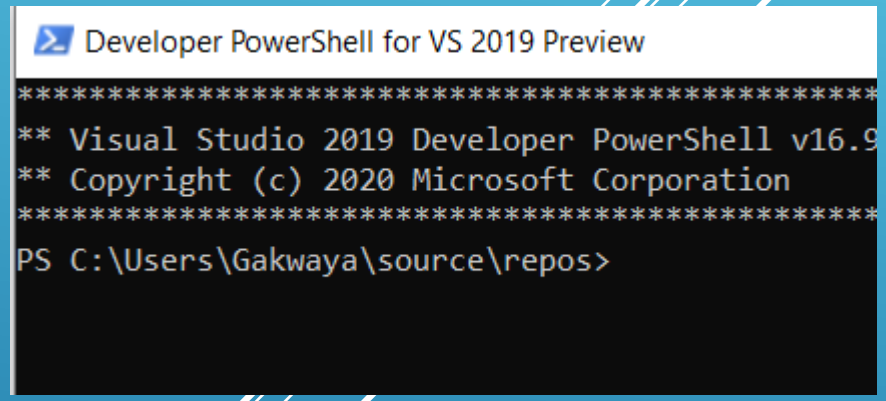
Debugging

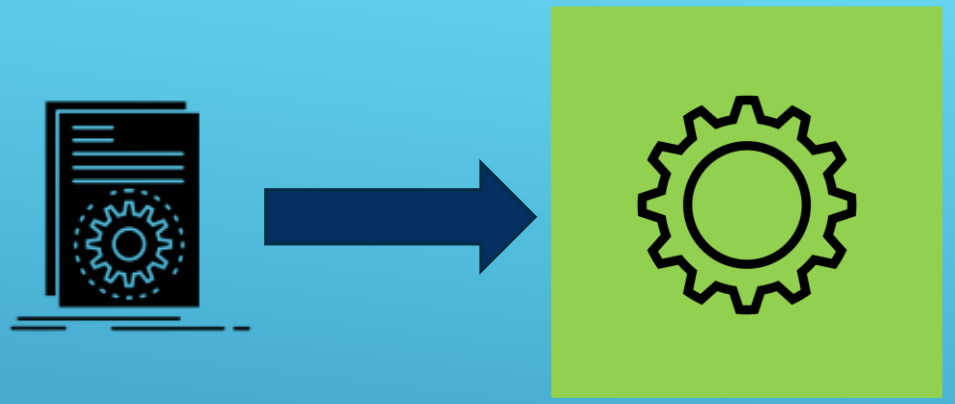
Running your program through some other program to make it freeze at some point. This gives the ability to execute it line by line, jumping into functions, and examining the local variables in the current stack activation record.

```cpp
#include <iostream>

int add_numbers(int a, int b)
{
    return a + b;
}

int main()
{
    int a = 10;
    int b = 5;
    int c;

    std::cout << "Statement1" << std::endl;
    std::cout << "Statement2" << std::endl;
    c = add_numbers(a, b);
    std::cout << "Statement3" << std::endl;
    std::cout << "Statement4" << std::endl;

    return 0;
}
```

Compiler

```
a = 10      (int)
b = 5       (int)
c           (int)
print("Statement1")
print("Statement2")
c = f_add(a,b)
print("Statement3")
print("Statement4")
end
```

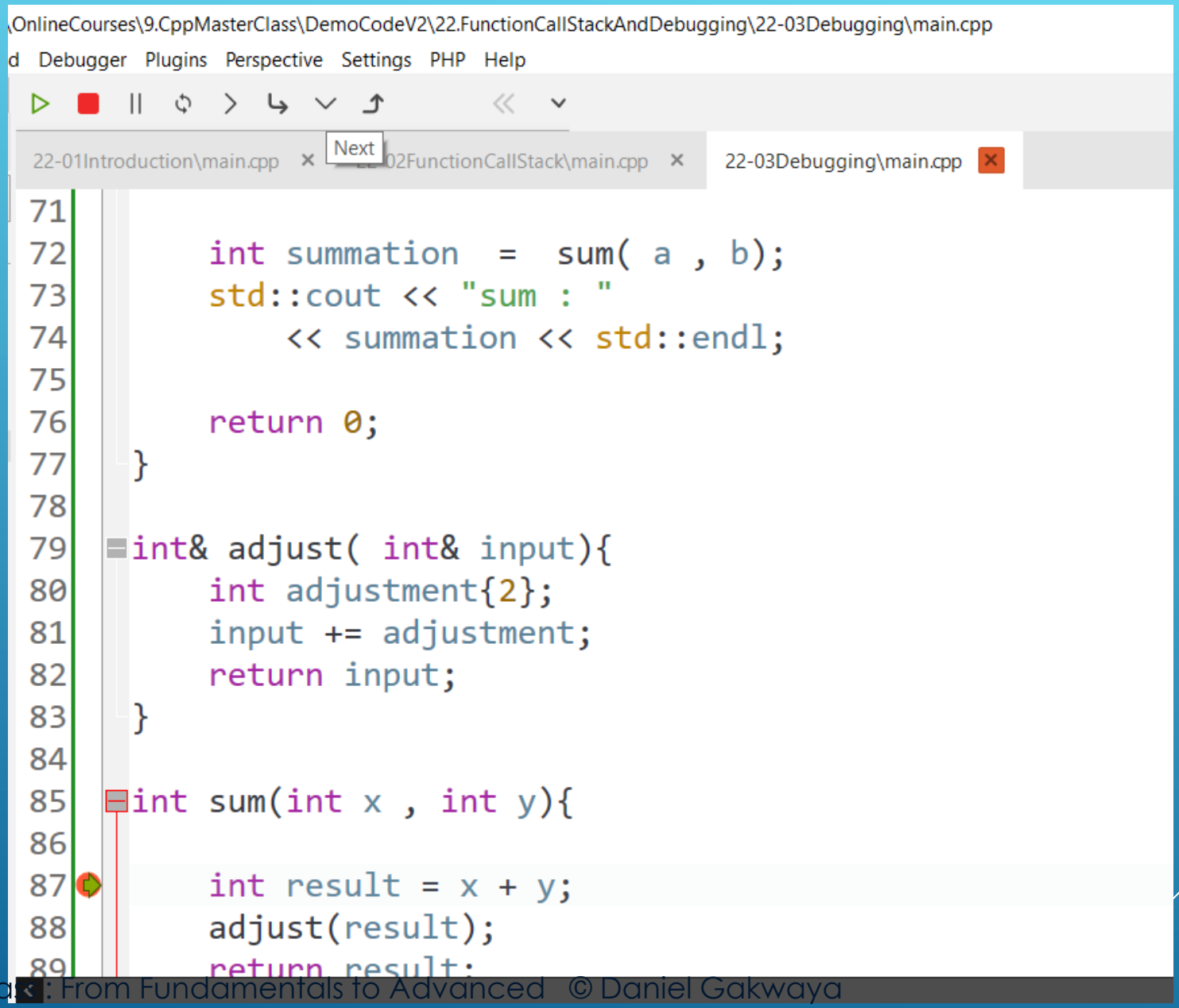

Developer PowerShell for VS 2019 Preview
****
** Visual Studio 2019 Developer PowerShell v16.9
** Copyright (c) 2020 Microsoft Corporation
****
PS C:\Users\Gakwaya\source\repos>



```
Developer PowerShell for VS 2019 Preview
**********************************************
** Visual Studio 2019 Developer PowerShell v16.9
** Copyright (c) 2020 Microsoft Corporation
**********************************************
PS C:\Users\Gakwaya\source\repos>
```

22-01Introduction\main.cpp | 22-02FunctionCallStack\main.cpp | 22-03Debugging\main.cpp

```cpp
int main(int argc, char **argv){

    int a{10};
    int b{12};

    int summation  =  sum( a , b);
    std::cout << "sum : "
        << summation << std::endl;

    return 0;
}

int& adjust( int& input){
    int adjustment{2};
    input += adjustment;
    return input;
}

int sum(int x , int y){

    int result = x + y;
    adjust(result);
    return result;
}
```


\OnlineCourses\9.CppMasterClass\DemoCodeV2\22.FunctionCallStackAndDebugging\22-03Debugging\main.cpp
d Debugger Plugins Perspective Settings PHP Help
Next
22-01Introduction\main.cpp
02FunctionCallStack\main.cpp
22-03Debugging\main.cpp
71
72 int summation = sum(a , b);
73 std::cout << "sum : "
74 << summation << std::endl;
75
76 return 0;
77 }
78
79 int& adjust(int& input){
80 int adjustment{2};
81 input += adjustment;
82 return input;
83 }
84
85 int sum(int x , int y){
86
87 int result = x + y;
88 adjust(result);
89 return result;


Locals
Watches
Call Stack

- Visual Studio Code
- CodeLite
- Microsoft Visual Studio

Slide intentionally left empty

Debugging in Visual Studio Code

```cpp
int main(int argc, char **argv){

    int a{10};
    int b{12};

    int summation  =  sum( a , b);
    std::cout << "sum : "
        << summation << std::endl;

    return 0;
}

int& adjust( int& input){
    int adjustment{2};
    input += adjustment;
    return input;
}

int sum(int x , int y){

    int result = x + y;
    adjust(result);
    return result;
}
```


Locals
Watches
Call Stack

Documentation

Run/Debug configurations in launch.json files

Debug tools from gcc

```
8       {
9           "name": "g++.exe - Build and debug active file",
10          "type": "cppdbg",
11          "request": "launch",
12          "program": "${fileDirname}\\rooster.exe",
13          "args": [],
14          "stopAtEntry": false,
15          "cwd": "${workspaceFolder}",
16          "environment": [],
17          "externalConsole": true,
18          "MIMode": "gdb",
19          "miDebuggerPath": "C:\\mingw64\\bin\\gdb.exe",
20          "setupCommands": [
21              {
22                  "description": "Enable pretty-printing for gdb",
23                  "text": "-enable-pretty-printing",
24                  "ignoreFailures": true
25              }
26          ],
27          "preLaunchTask": "Build with GCC 10.0.2 Mingw"
28      },
```

Debug tools from the Microsoft C++ compiler

```
29          {
30              "name": "cl.exe build and debug active file",
31              "type": "cppvsdbg",
32              "request": "launch",
33              "program": "${fileDirname}\\rooster.exe",
34              "args": [],
35              "stopAtEntry": false,
36              "cwd": "${workspaceFolder}",
37              "environment": [],
38             "console": "externalTerminal",
39              "preLaunchTask": "Build with MSVC"
40          }
```

Slide intentionally left empty

Debugging In CodeLite

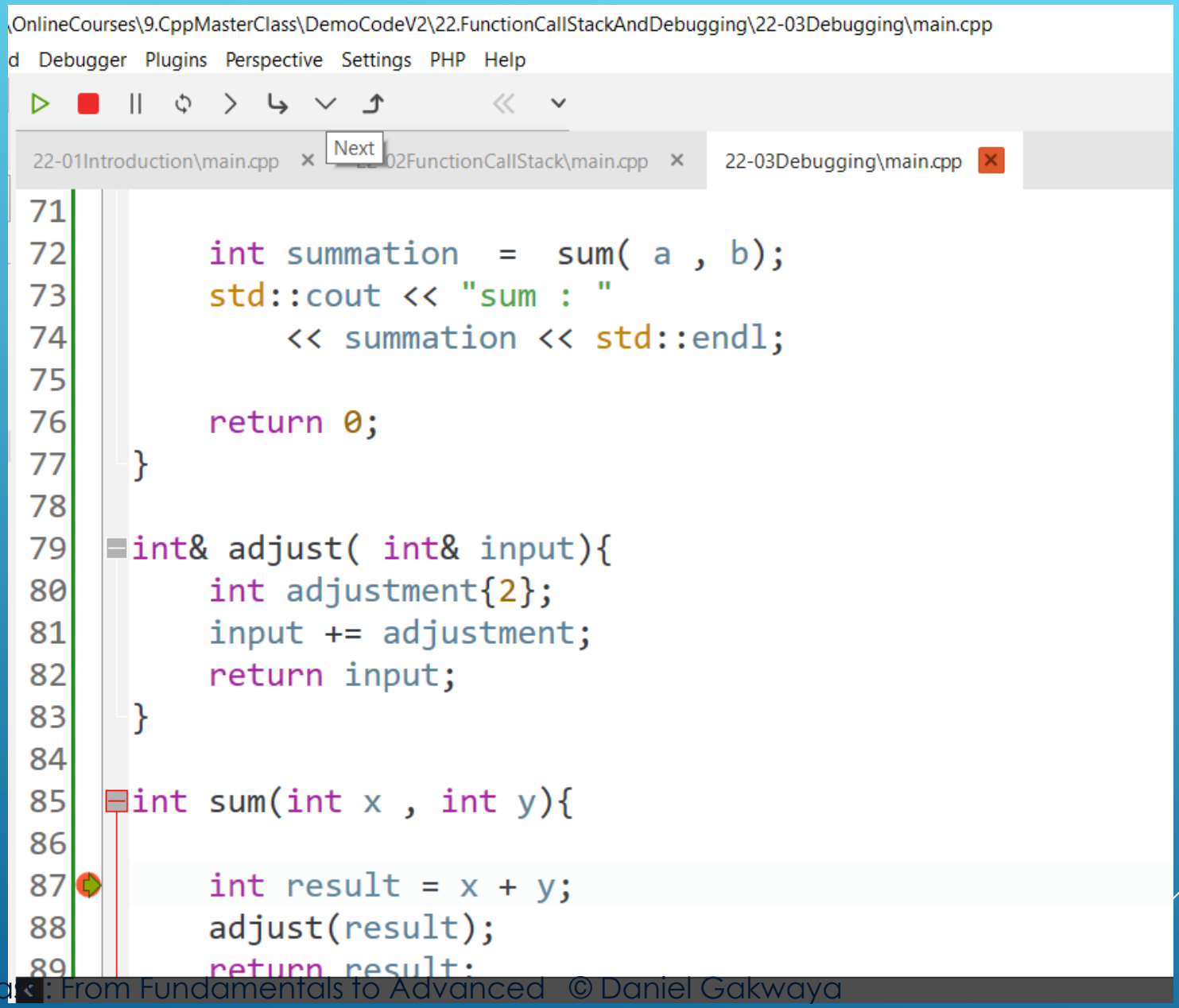
Debugging

Running your program through some other program to make it freeze at some point. This gives the ability to execute it line by line, jumping into functions, and examining the local variables in the current stack activation record.

Break points

22-01Introduction\main.cpp × 22-02FunctionCallStack\main.cpp × 22-03Debugging\main.cpp ×

```
67  int main(int argc, char **argv){
68
69      int a{10};
70      int b{12};
71
72      int summation  =  sum( a , b);
73      std::cout << "sum : "
74          << summation << std::endl;
75
76      return 0;
77  }
78
79  int& adjust( int& input){
80      int adjustment{2};
81      input += adjustment;
82      return input;
83  }
84
85  int sum(int x , int y){
86
87      int result = x + y;
88      adjust(result);
89      return result;
90  }
```


\OnlineCourses\9.CppMasterClass\DemoCodeV2\22.FunctionCallStackAndDebugging\22-03Debugging\main.cpp
d Debugger Plugins Perspective Settings PHP Help
Next
22-01Introduction\main.cpp
02FunctionCallStack\main.cpp
22-03Debugging\main.cpp
71
72 int summation = sum(a , b);
73 std::cout << "sum : "
74 << summation << std::endl;
75
76 return 0;
77 }
78
79 int& adjust(int& input){
80 int adjustment{2};
81 input += adjustment;
82 return input;
83 }
84
85 int sum(int x , int y){
86
87 int result = x + y;
88 adjust(result);
89 return result;


Locals
Watches
Call Stack

Slide intentionally left empty

Debugging in Microsoft Visual Studio

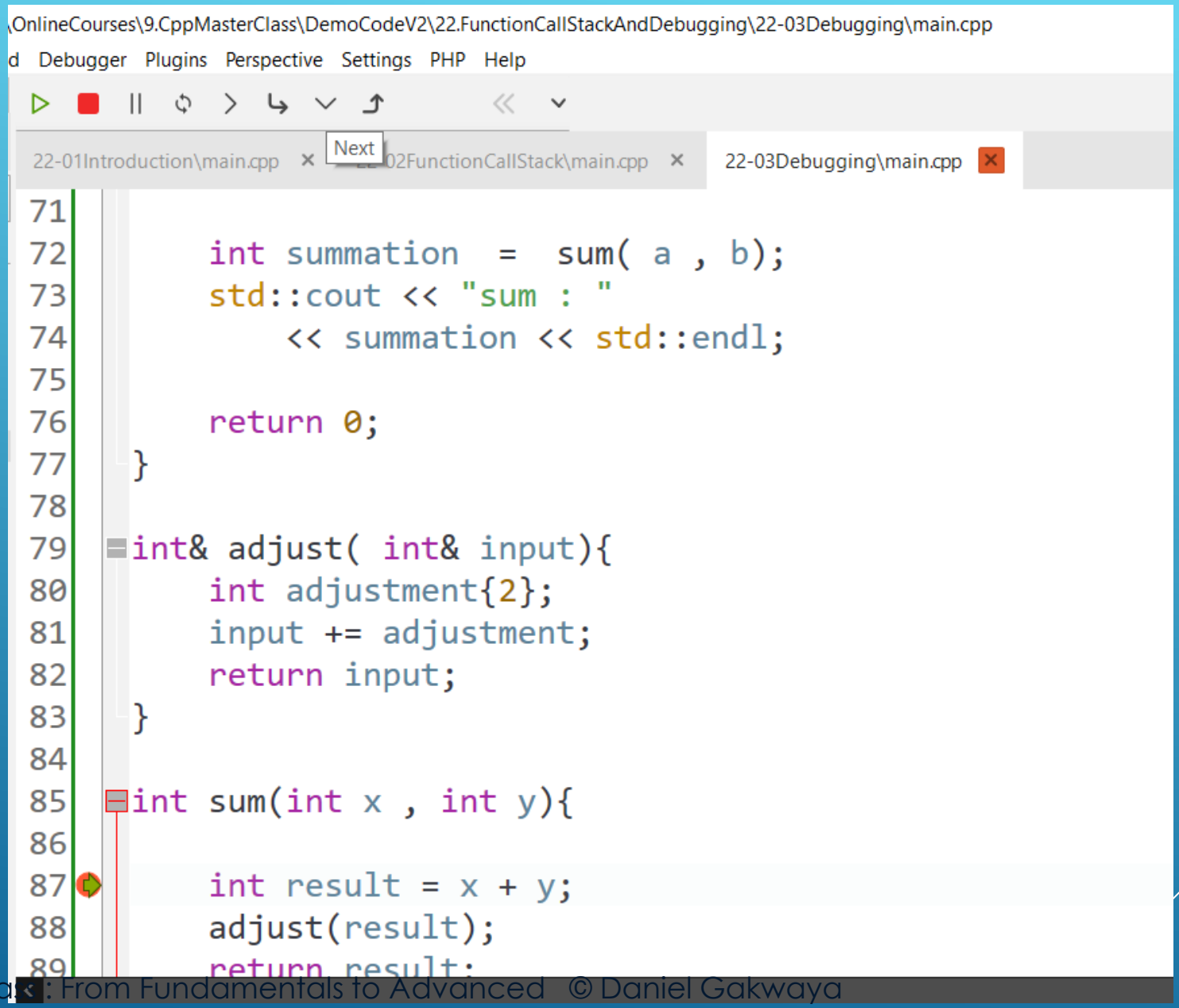
Debugging

Running your program through some other program to make it freeze at some point. This gives the ability to execute it line by line, jumping into functions, and examining the local variables in the current stack activation record.

Break points

22-01Introduction\main.cpp × 22-02FunctionCallStack\main.cpp × 22-03Debugging\main.cpp ×

```
67  int main(int argc, char **argv){
68
69      int a{10};
70      int b{12};
71
72      int summation  =  sum( a , b);
73      std::cout << "sum : "
74          << summation << std::endl;
75
76      return 0;
77  }
78
79  int& adjust( int& input){
80      int adjustment{2};
81      input += adjustment;
82      return input;
83  }
84
85  int sum(int x , int y){
86
87      int result = x + y;
88      adjust(result);
89      return result;
90  }
```


\OnlineCourses\9.CppMasterClass\DemoCodeV2\22.FunctionCallStackAndDebugging\22-03Debugging\main.cpp
d Debugger Plugins Perspective Settings PHP Help
Next
22-01Introduction\main.cpp
02FunctionCallStack\main.cpp
22-03Debugging\main.cpp
71
72 int summation = sum(a , b);
73 std::cout << "sum : "
74 << summation << std::endl;
75
76 return 0;
77 }
78
79 int& adjust(int& input){
80 int adjustment{2};
81 input += adjustment;
82 return input;
83 }
84
85 int sum(int x , int y){
86
87 int result = x + y;
88 adjust(result);
89 return result;


Locals
Watches
Call Stack

Slide intentionally left empty

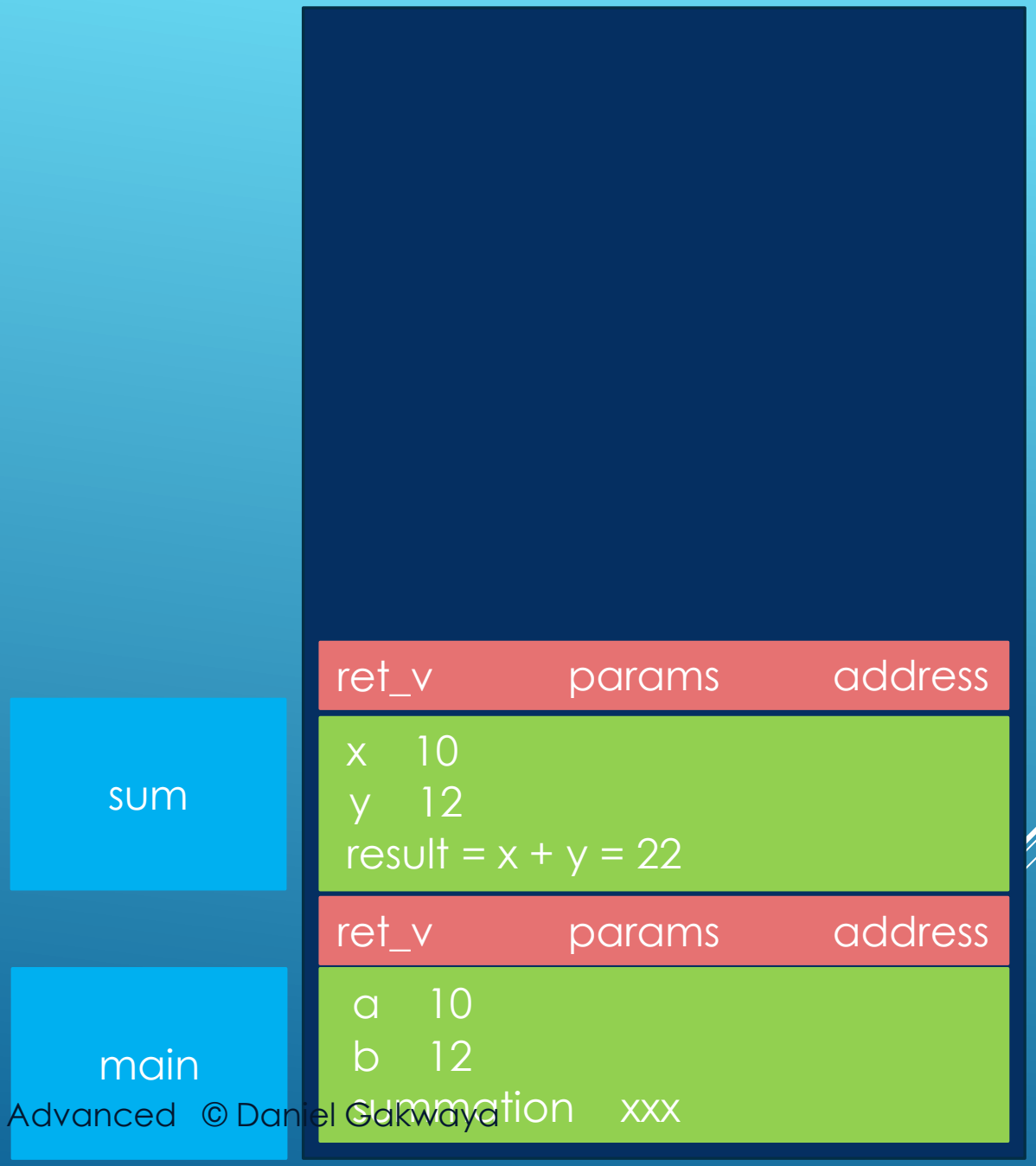
Debugging arrays, loops and pointers

```cpp
#include <iostream>

double sum ( double array[], size_t count){

    double sum{};
    for(size_t i{} ; i < count ; ++i){
        sum += array[i];
    }
    return sum;

}


int main(int argc, char **argv)
{
    double numbers[] {10.0,20.0,30.0,40.0,50.0}; // Sum should be 150.0

    double total = sum(numbers,std::size(numbers));

    std::cout << "sum : " << total << std::endl;

    return 0;
}
```

Slide intentionally left empty

Function call stack & Debugging : Summary

```cpp
int main(int argc, char **argv){

    int a{10};
    int b{12};

    int summation  =  sum( a , b);
    std::cout << "sum : "
        << summation << std::endl;

    return 0;
}

int& adjust( int& input){
    int adjustment{2};
    input += adjustment;
    return input;
}

int sum(int x , int y){

    int result = x + y;
    adjust(result);
    return result;
}
```



- Visual Studio Code
- CodeLite
- Microsoft Visual Studio

Slide intentionally left empty