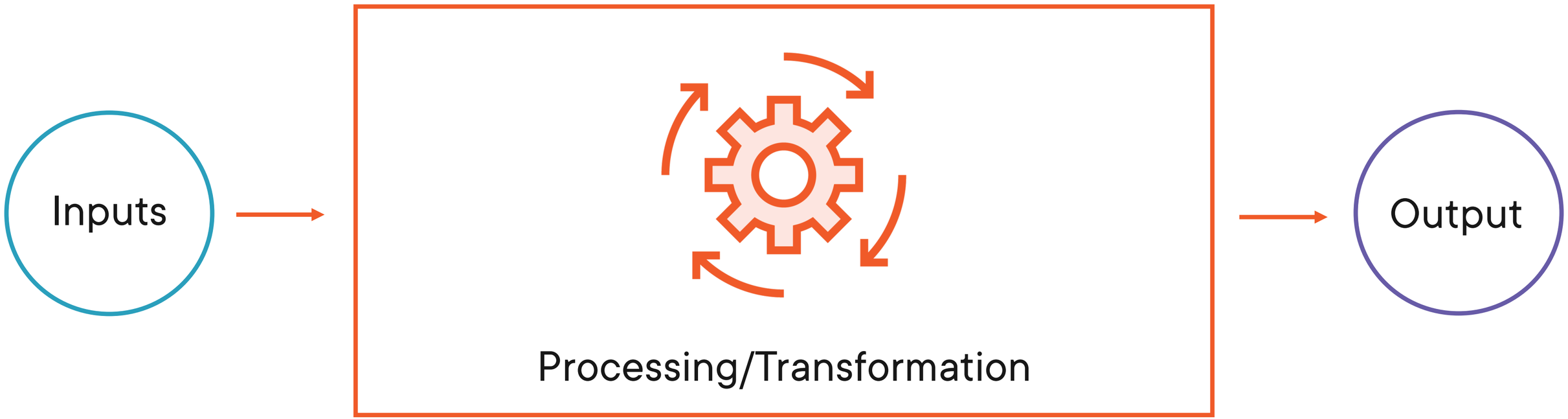
Working with Functions in C

Understanding Functions in C

Zachary Bennett
Software Engineer

@z_bennett_ zachbennettcodes.com

What are Functions?



C Function Groups

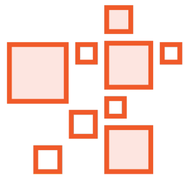
User-defined

Functions that are created by the programmer using C and its library functions

Library

Functions that are included with the C programming language

Benefits to Functions



Modularity

Reusable code

Maintainability

Easy to test systems

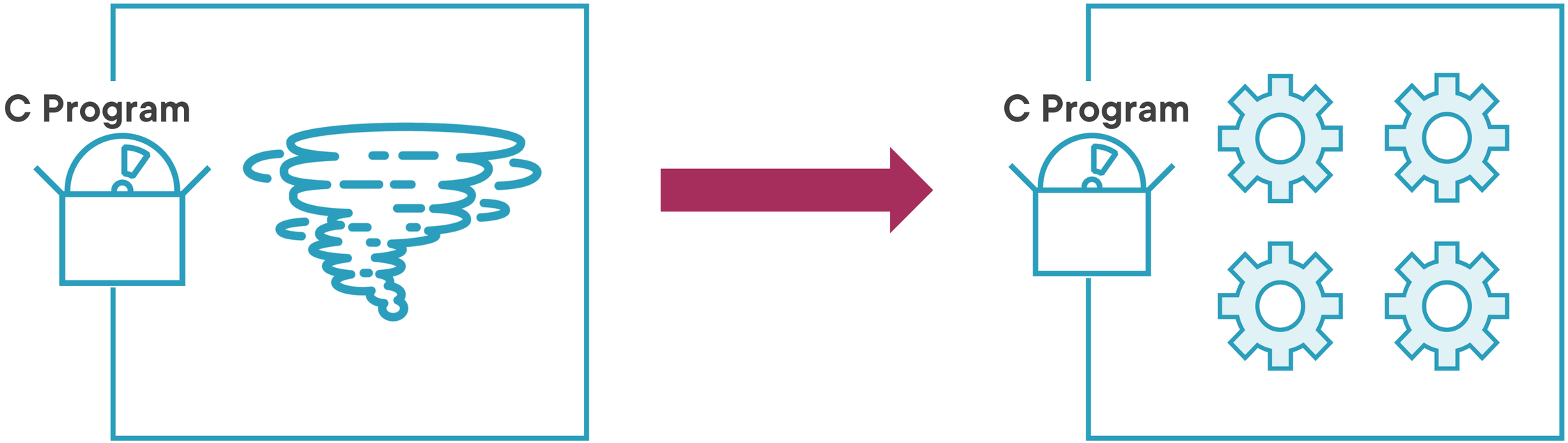
Less bugs

Why Functions?

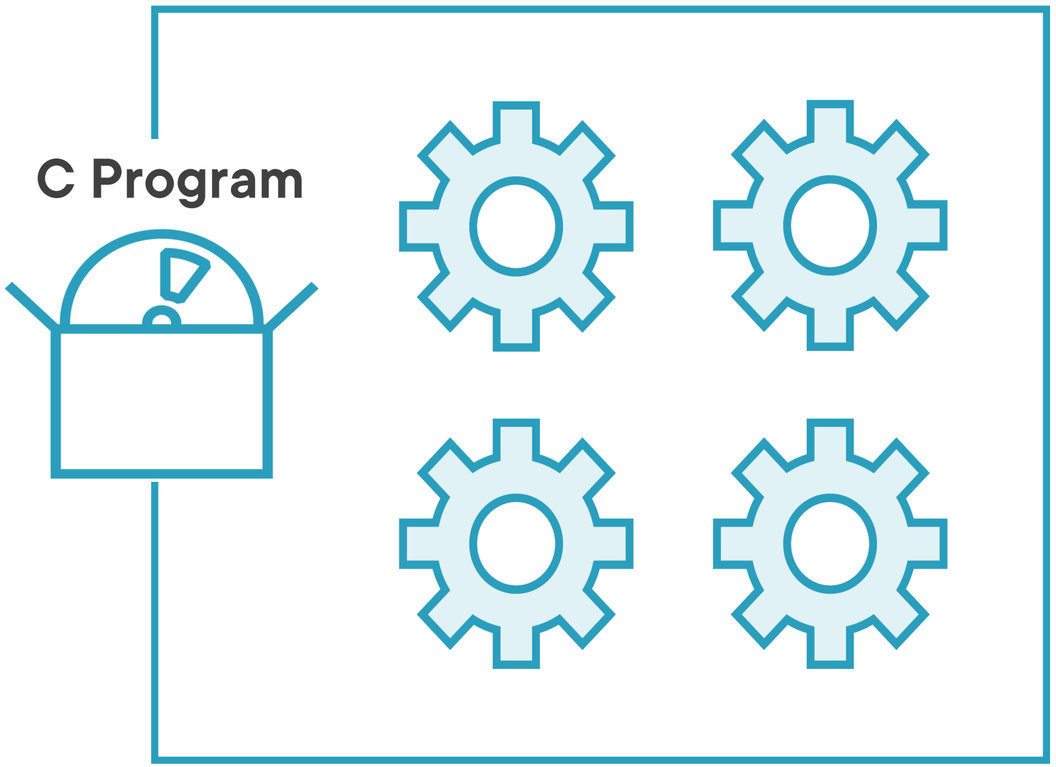
```
main() {
    ...processing step one...
    ...processing step two...
    ...processing step three...
}
```

- Define steps separately
- Easier to test each step
- Quicker to debug

Modular and Maintainable Code

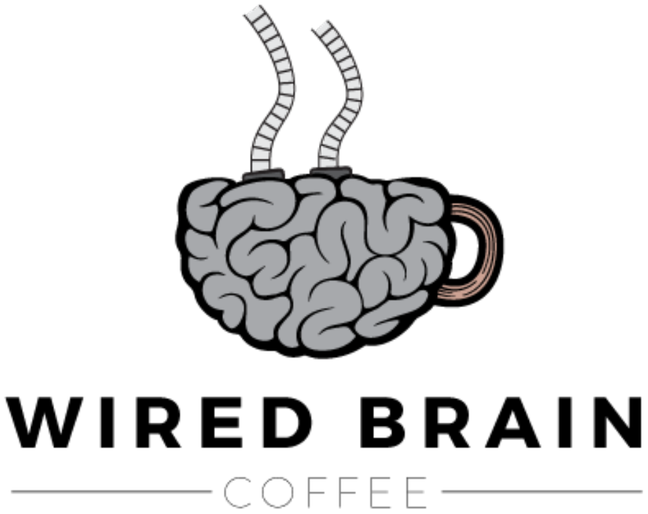


Ease of Testing




```
int main() {
  RUN_TEST(function_a_works)
  RUN_TEST(function_b_works)
  RUN_TEST(function_c_works)
  RUN_TEST(function_d_works)
}
```

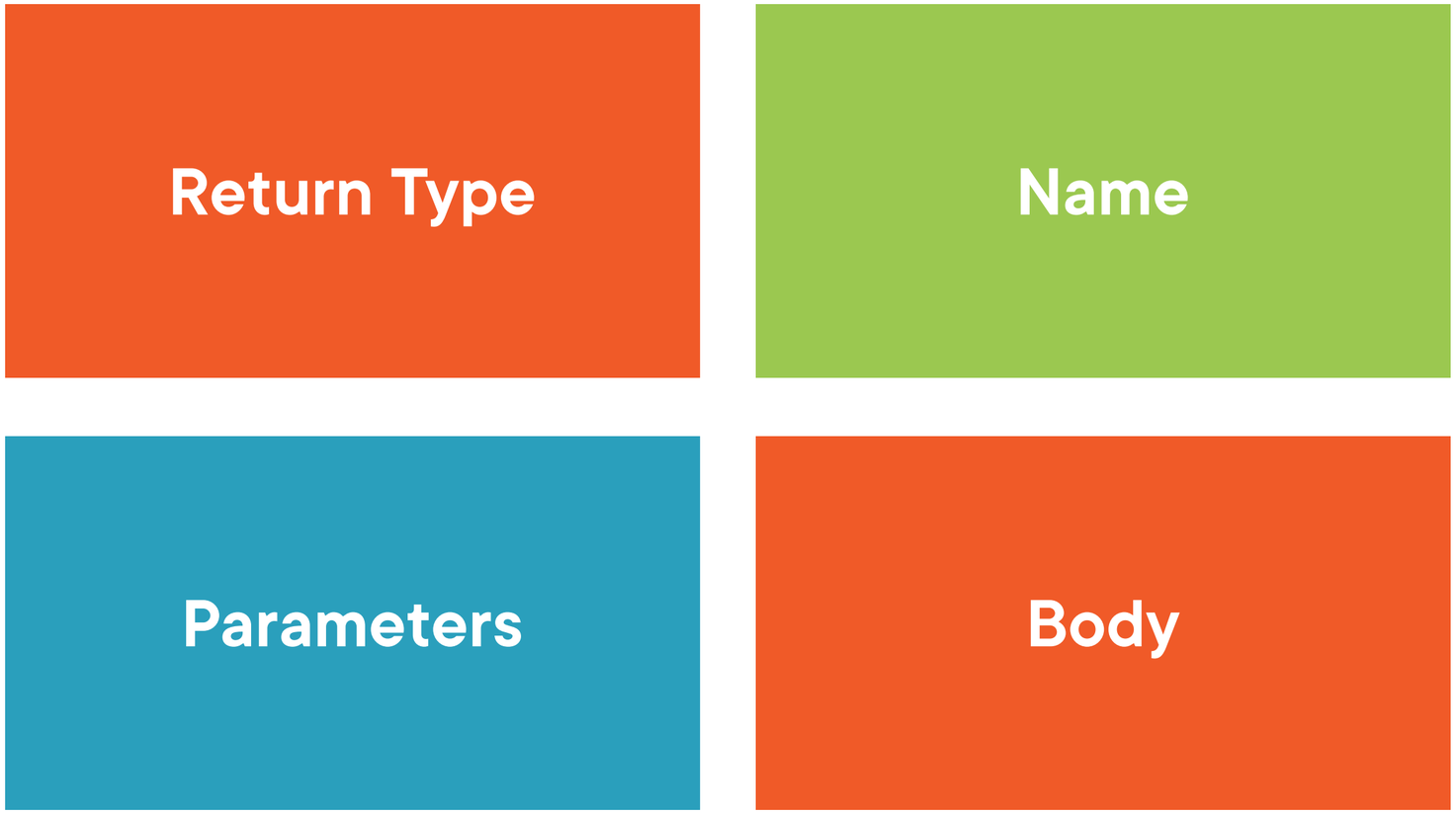
100% Code Coverage



Wired Brain Coffee Needs Help

- Improve C coding style
- Modularity and maintainability

Anatomy of a Function in C



```c
int main(int argc, char* argv[]) {

    char* my_name = “zach”;

    printf(”%s”, my_name);


    return 0;

}
```

Inspecting the “main” Function

The four, basic components of a function in C can be seen color-coded above. These components consist of the return type, the function name, the function parameters and the function body.

```
// Example user-defined function
void alloc_string(int size, char** out) {

  // Set pointer managed by caller
  *out = malloc(sizeof(char) * size + 1);
}

// Example Usage
char* my_ptr = NULL;
alloc_string(5, &my_ptr);
```

- Return type is “void” meaning nothing is returned and function name is “alloc_string”
- Function parameters consist of a size and a pointer to a character pointer
- The function body dereferences the “out” pointer and sets its inner pointer to the returned value of the call to malloc
- Example of how to call the alloc_string function with the necessary function arguments

Parameters vs. Arguments

Function Parameter

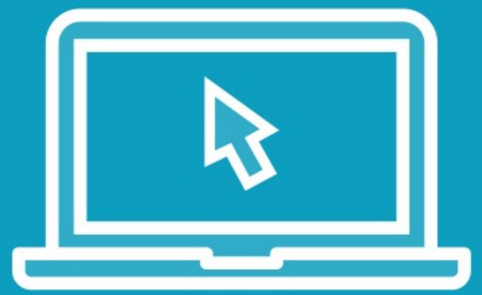
Part of a function's "signature" that is relevant to the function's definition

Function Argument

The function arguments are the values you use when calling the function

Up Next:
Demo: Creating and Calling Functions

Demo



- Inspecting a function by its parts
- Calling a function
  - Call the function improperly
  - Get help from the compiler/linter
  - Call the function correctly

Summary

- Function anatomy – four basic parts
  - Return Type
  - Name
  - Parameters
  - Body
- Difference between parameters and arguments
- Calling functions
  - Dissecting a functions parts to understand it
  - Receive help from the compiler!

Up Next:
User-defined Functions