In C functions that return complex types or arrays must necessarily return pointers to memory that has been allocated on the program *heap* rather than the program *stack*. The reason for this is that the program stack space is reused by subsequent function calls and the contents may even be changed when a function returns control back to the calling function (sometimes called *unwinding*). Thus, variables that are local in scope cannot be presumed to be available upon returning from a function. This is not an issue when returning primitive types as the return value is placed in a special memory location available to the calling function which presumably takes the value from that location and copies it to a variable that is within the scope of a calling function.

An illustrative example of this process can be found in Code Sample 1 and Figure 1.

Code Sample 1: Example returning a static array

```
#include<stdlib.h>
#include<stdio.h>

int * foo(int n) {
  int i;
  int b[5];
  for(i=0; i<5; i++) {
    b[i] = n*i;
  }
  for(i=0; i<5; i++) {
    printf("b[%d] = %d\n", i, b[i]);
  }
  return b;
}

int main(int argc, char **argv) {
  int i, m = 7;
  int *a = foo(m);
  for(i=0; i<5; i++) {
    printf("a[%d] = %d\n", i, a[i]);
  }
  return 0;
}
```

Stack Frame	Variable	Address	Content
		⋮	⋮
	b[4]	0x5c44cb76	25
	b[3]	0x5c44cb72	20
	b[2]	0x5c44cb68	15
	b[1]	0x5c44cb64	10
	b[0]	0x5c44cb60	5
	i	0x5c44cb56	5
foo	n	0x5c44cb52	7
		⋮	⋮
	a	0x5c44cb34	NULL
	m	0x5c44cb30	7
main	i	0x5c44cb26	0

(a) Program stack at the end of the execution of `foo` prior to its returning control back to `main`.

Stack Frame	Variable	Address	Content
		⋮	⋮
	b[4]	0x5c44cb76	-626679356
	b[3]	0x5c44cb72	20
	b[2]	0x5c44cb68	15
	b[1]	0x5c44cb64	32767
	b[0]	0x5c44cb60	1564158624
	i	0x5c44cb56	5
foo	n	0x5c44cb52	7
		⋮	⋮
	a	0x5c44cb34	0x5c44cb60
	m	0x5c44cb30	7
main	i	0x5c44cb26	0

(b) Upon returning, the stack frame is no longer valid; `a` points to a stack memory address but the frame and its local variables are no longer valid. Some have been overwritten with other values. Subsequent usage or access of the values in `a` are undefined behavior.

Figure 1: Illustration of the pitfalls of returning a static array in C. Static arrays are locally scoped and exist only within the function/block in which they are declared. The program stack frame in which the variables are stored is invalid when the function returns control back to the calling function. Depending on how the system/compiler/language handles this *unwinding* process, values may be changed, unavailable, etc.