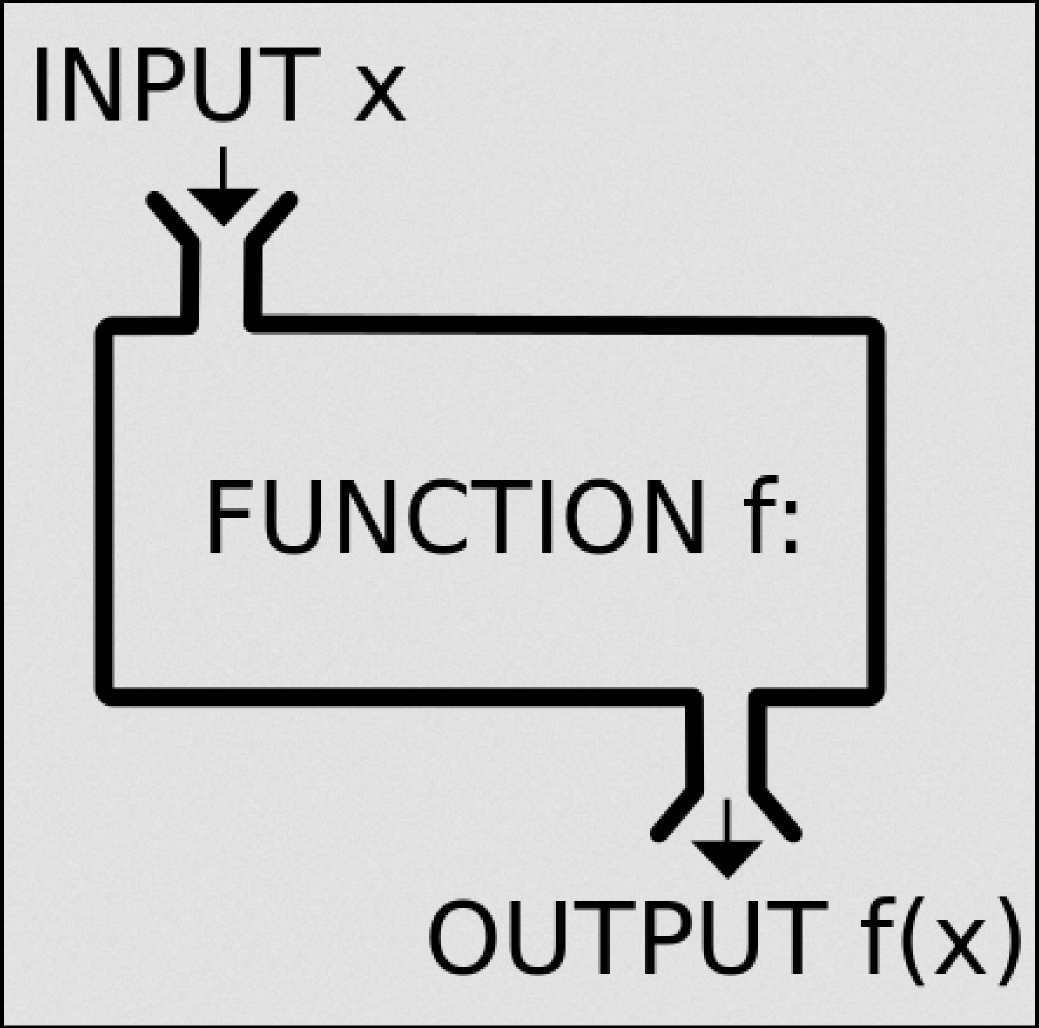
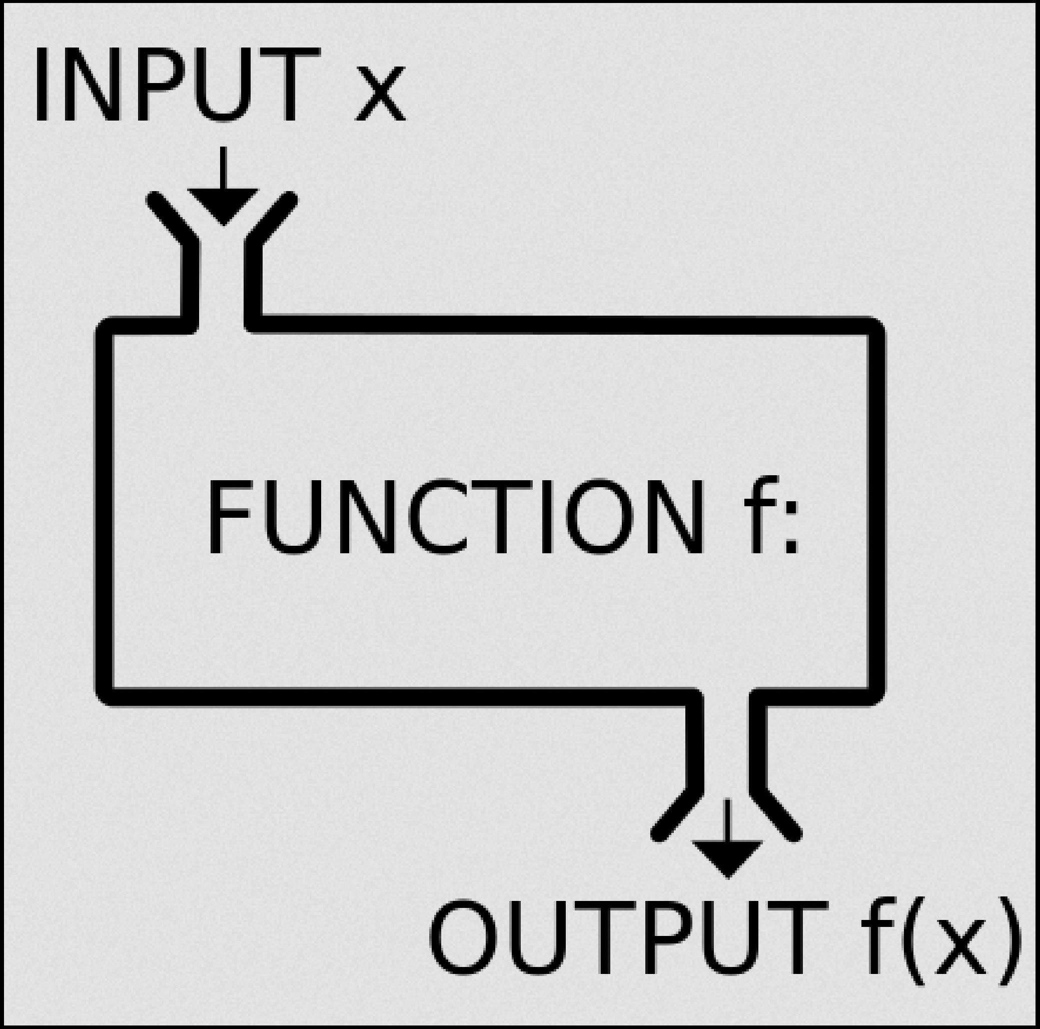
CSE102
Computer Programming



Credits: doval.scripts.mit.edu

What are Functions?

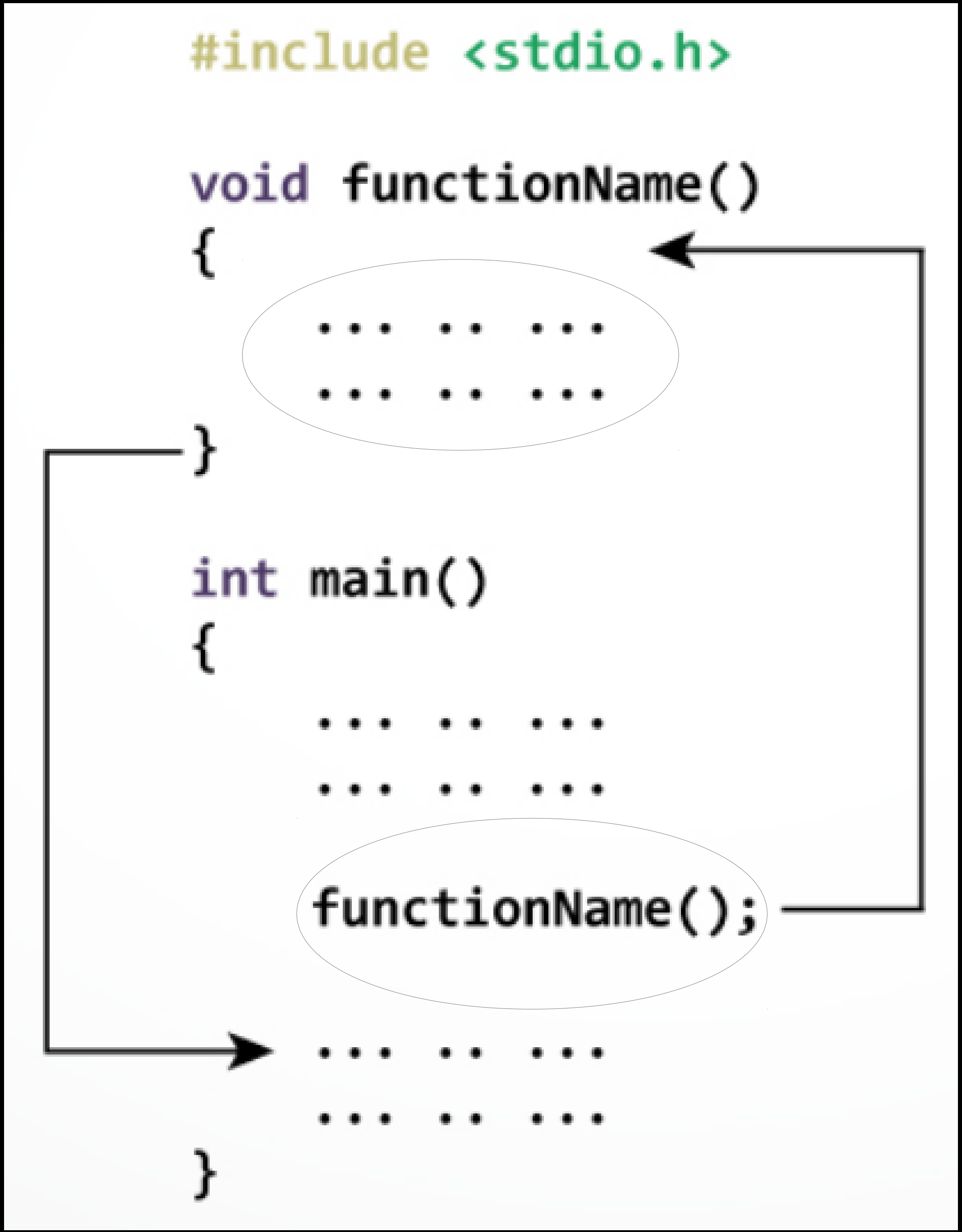
Also called methods, procedures etc.



Credits: doval.scripts.mit.edu

How Functions Work?

equivalent to having this code here

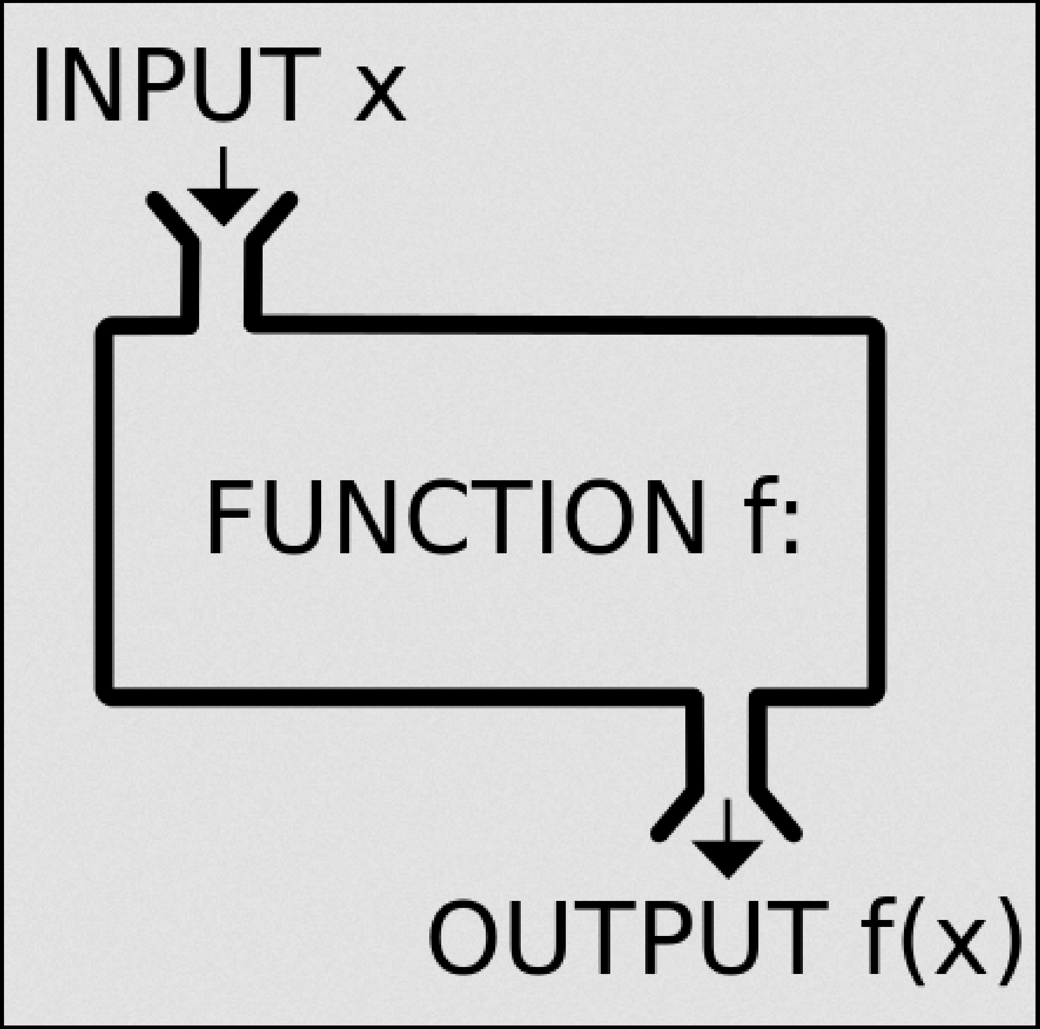


```
#include <stdio.h>

void functionName()
{
    ... .. ...
    ... .. ...
}

int main()
{
    ... .. ...
    ... .. ...

    functionName();

    ... .. ...
    ... .. ...
}
```

Functions are named piece of code

Credits: programiz.com

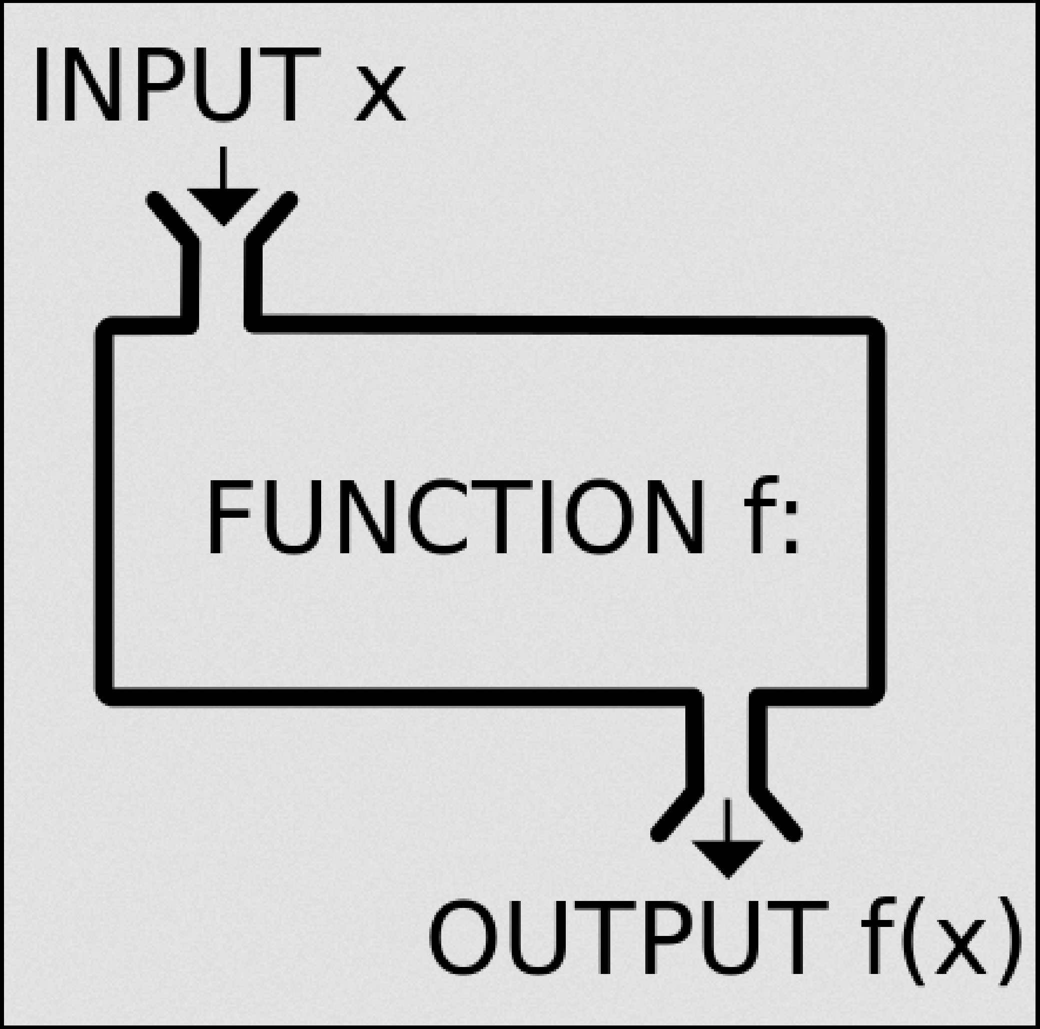
What Should They Have?



Credits: doval.scripts.mit.edu

What Should They Have?

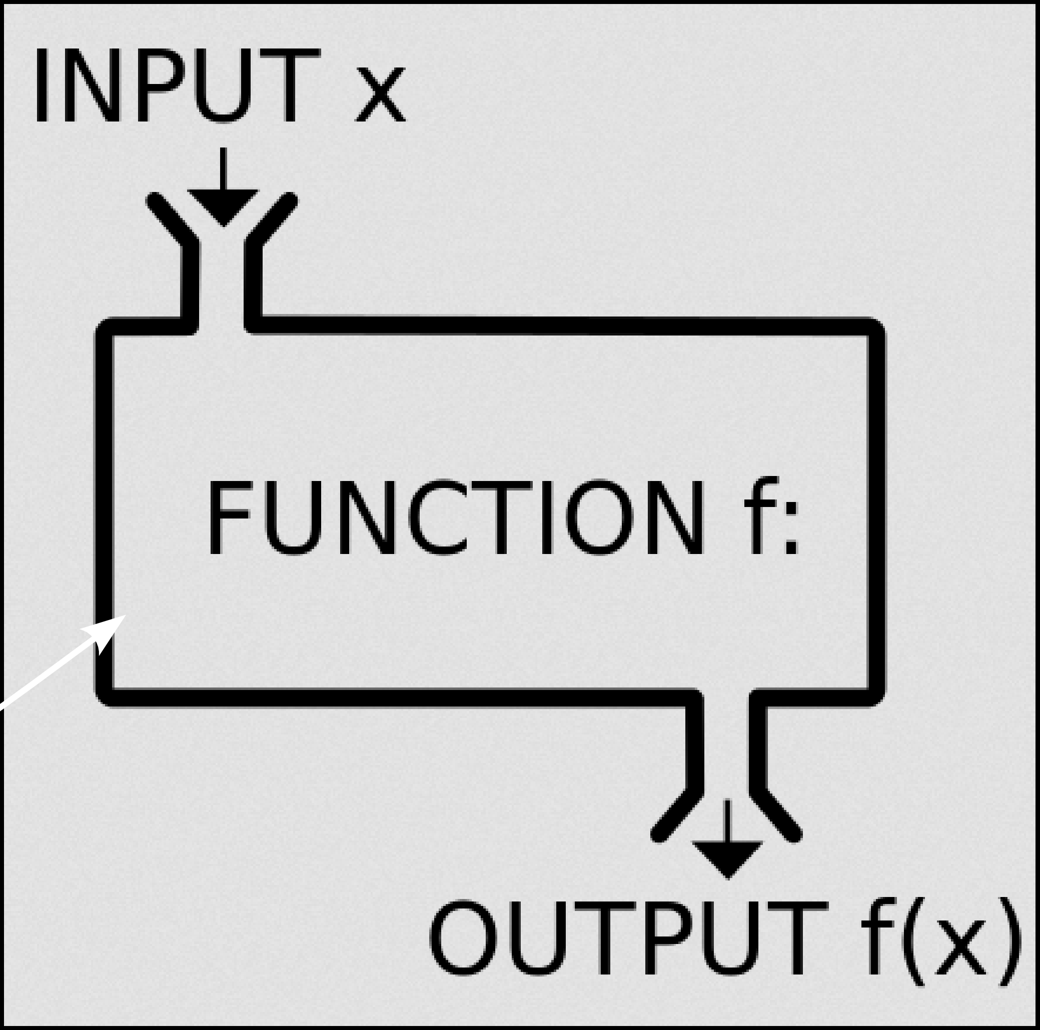
Should accept inputs (really huh!?)



What Should They Have?

Should accept inputs!

Should manipulate inputs

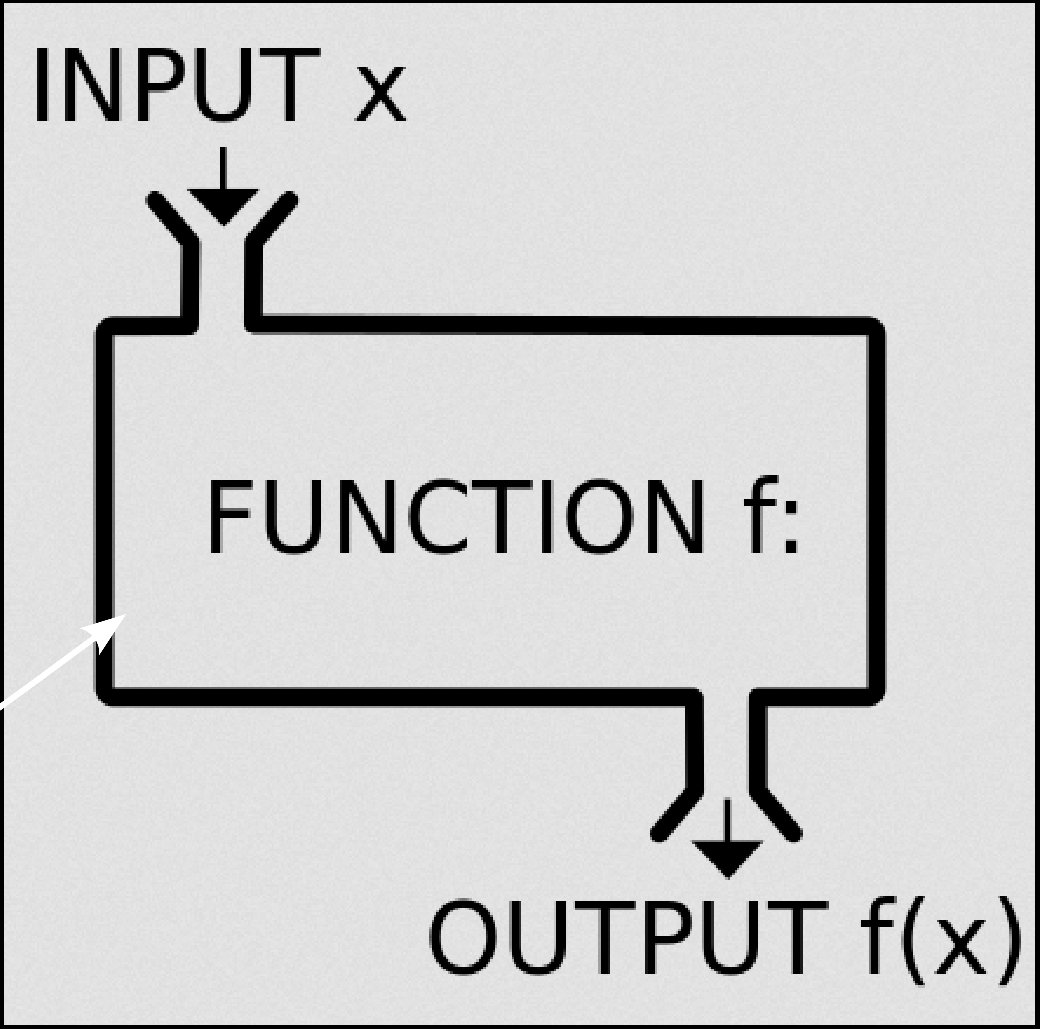


What Should They Have?

Should accept inputs!

Should manipulate inputs

Should return Outputs (really!)

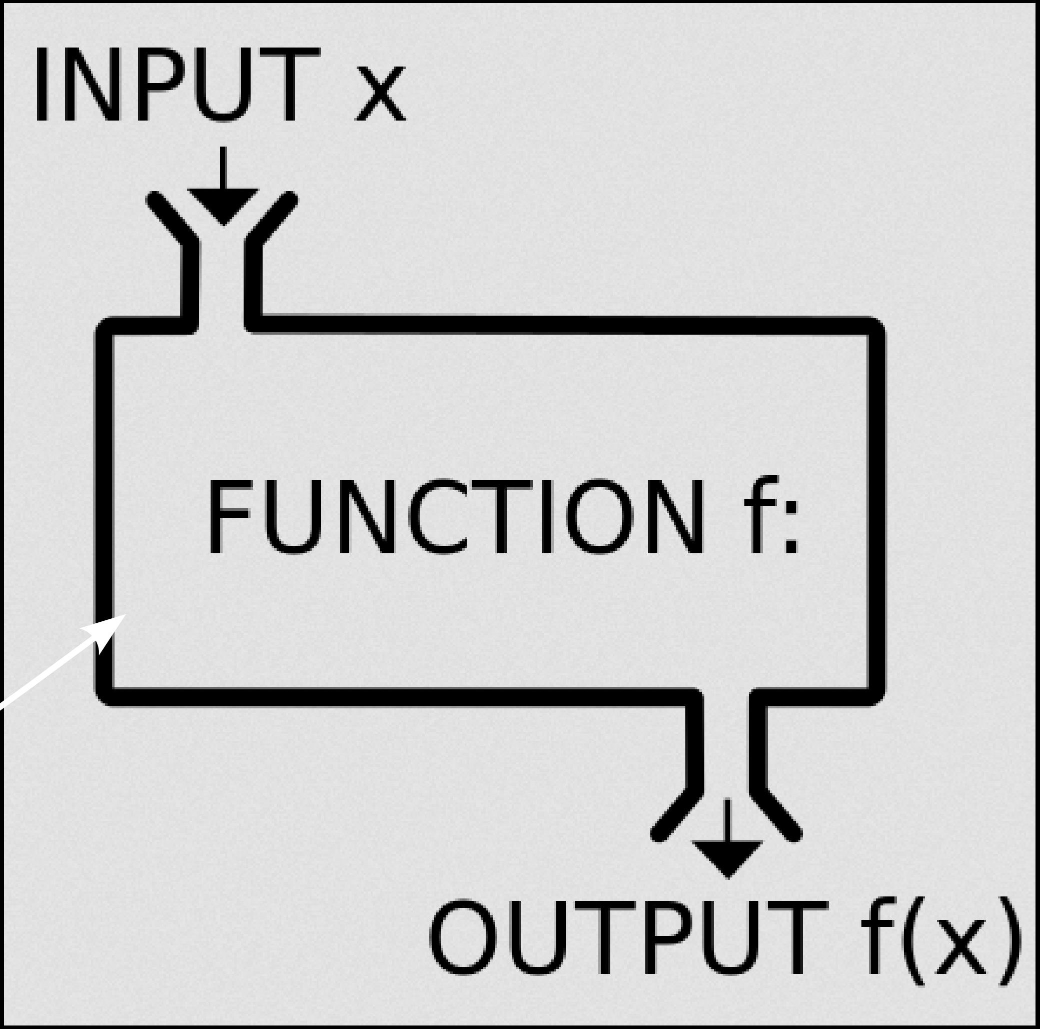


What Should They Have?

Should accept inputs!

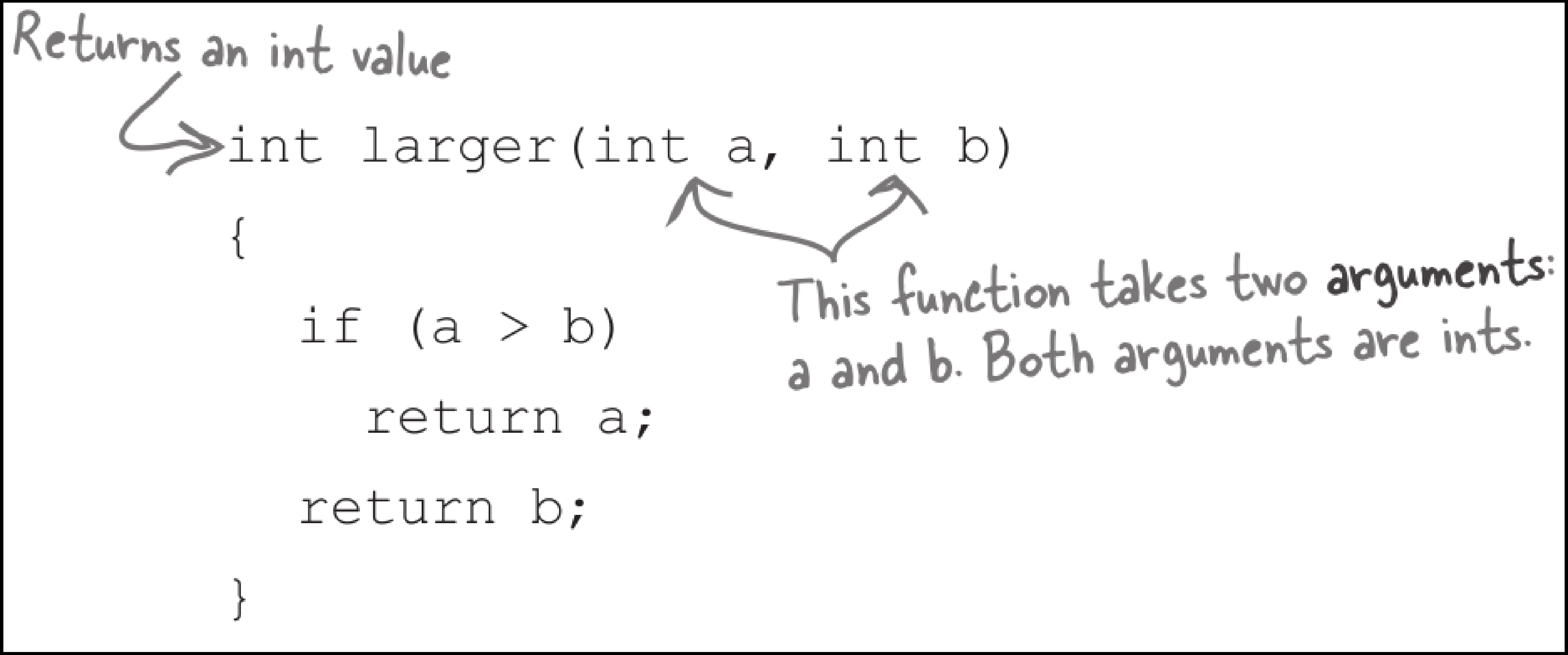
Should manipulate inputs

Should return outputs



Should have a name!!

How Do They Look Like?



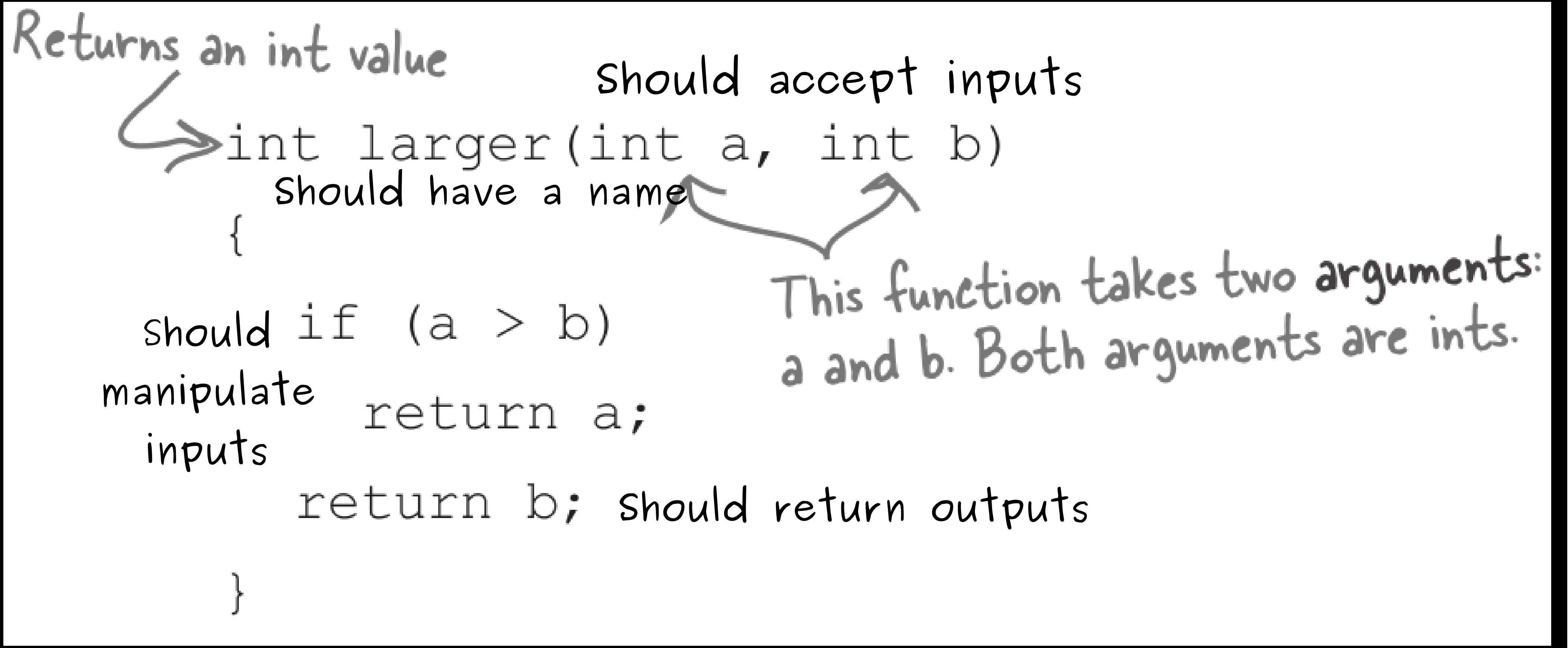
Returns an int value

```
int larger(int a, int b)
{
   if (a > b)
      return a;
   return b;
}
```

This function takes two **arguments**: a and b. Both arguments are ints.

Credits: Head First C

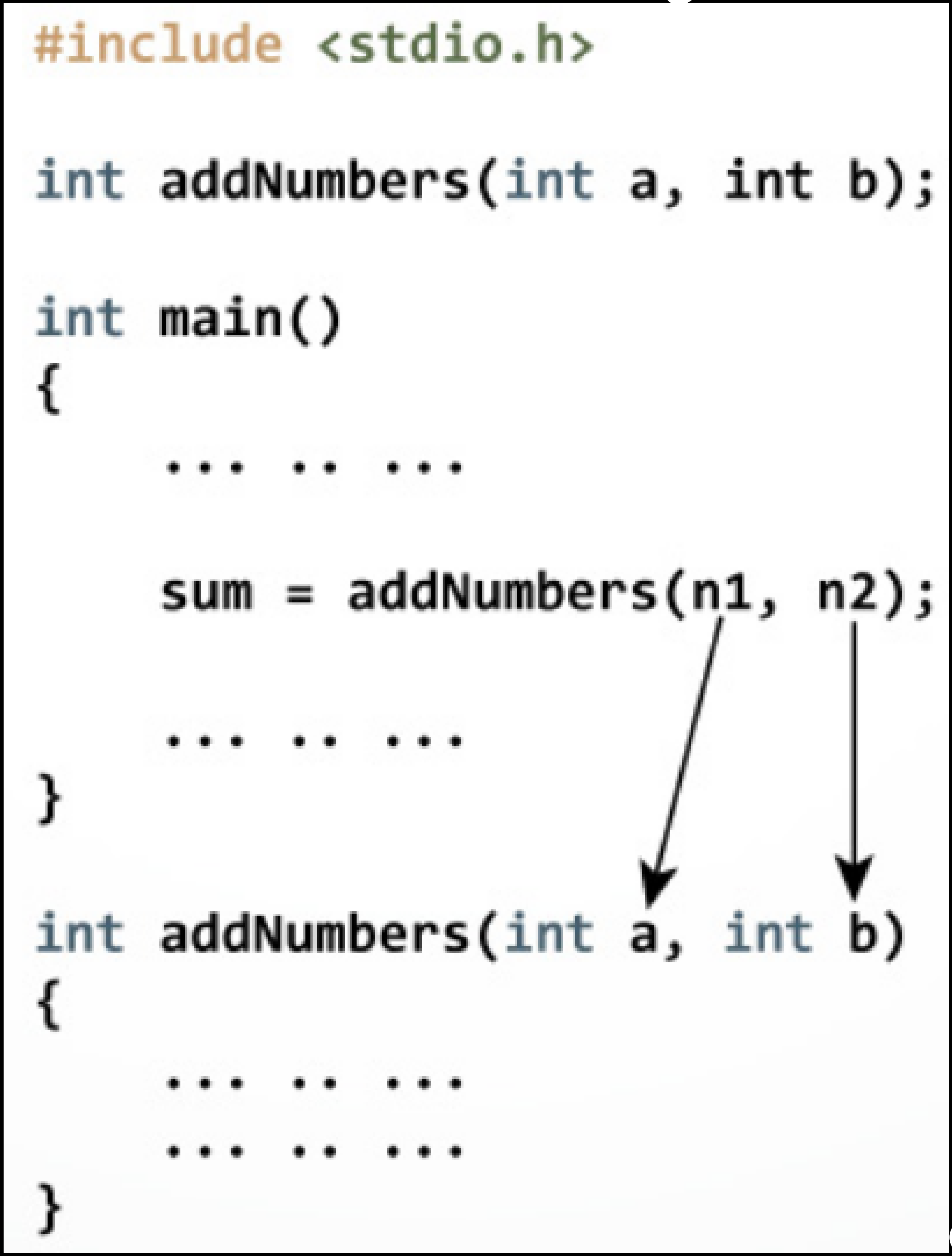
How Do They Look Like?



```
int larger(int a, int b)
{
    if (a > b)
        return a;
    return b;
}
```

Credits: Head First C

How are They Used?

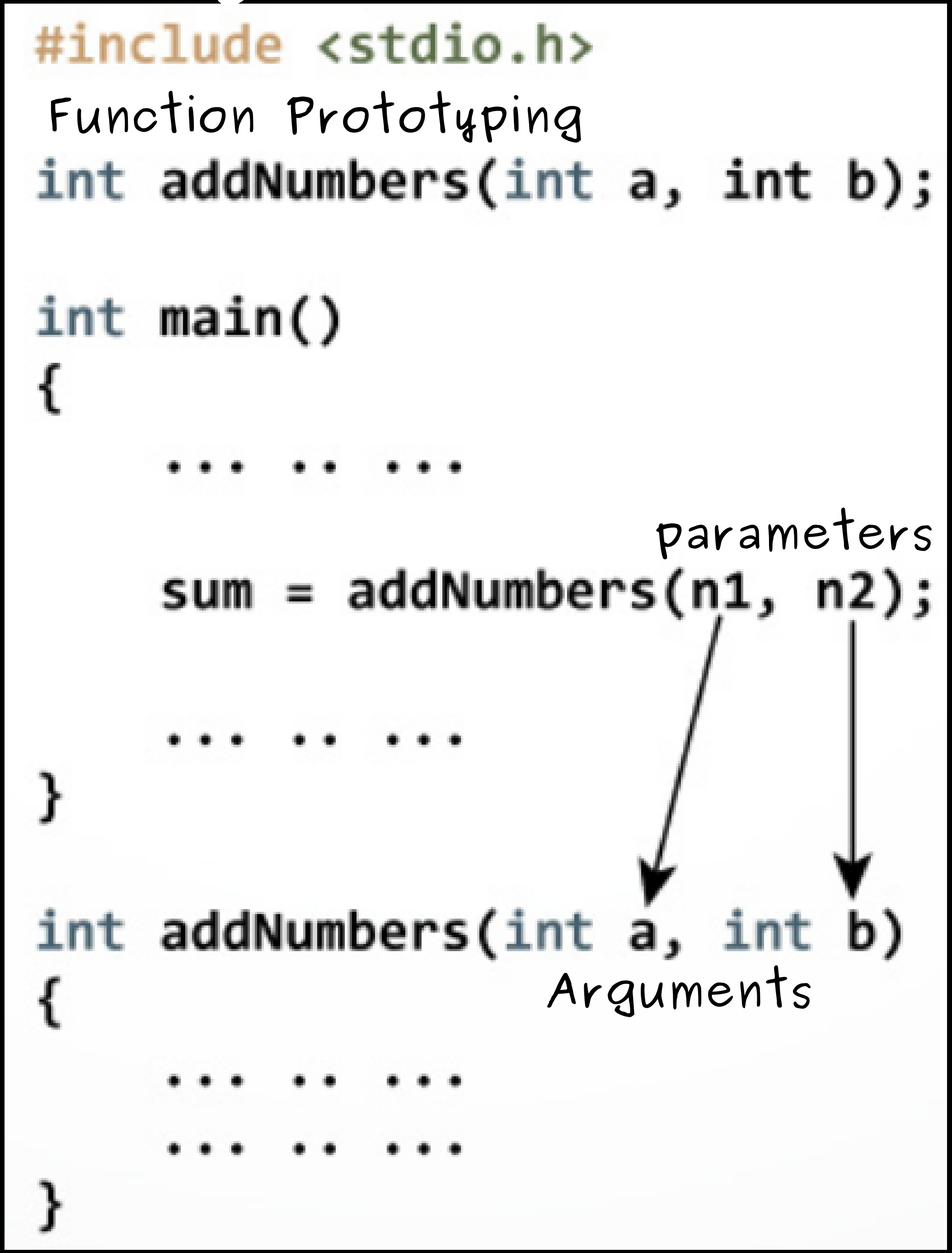


```
#include <stdio.h>

int addNumbers(int a, int b);

int main()
{
    ... .. ...

    sum = addNumbers(n1, n2);


    ... .. ...
}


int addNumbers(int a, int b)
{
    ... .. ...
    ... .. ...
}
```

Credits: programiz.com

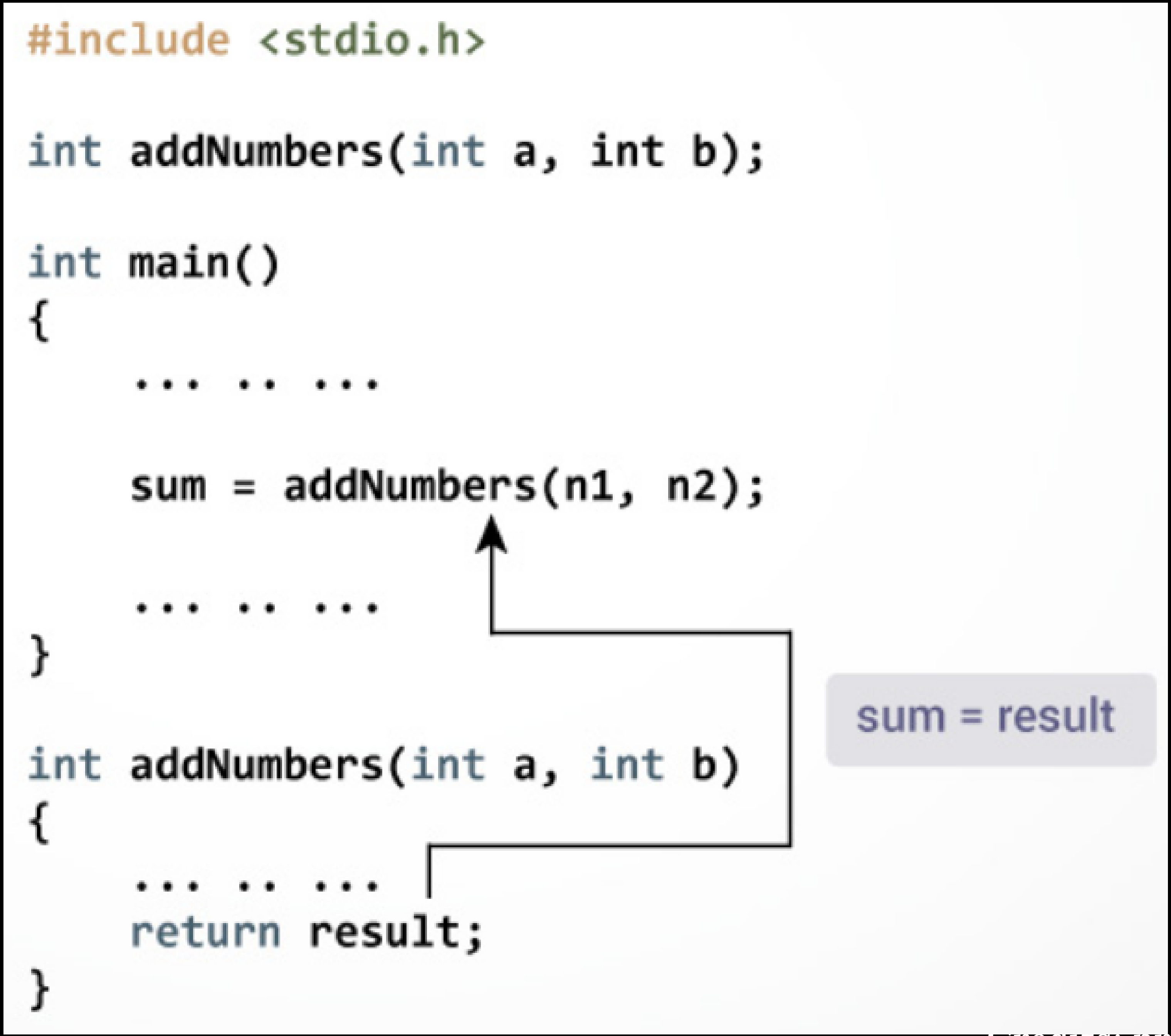
How are They Used?

Function prototyping declares all info. about function (name, no. Of args, return type) to the compiler like variable declaration



```
#include <stdio.h>
int addNumbers(int a, int b);

int main()
{
    ... .. ...

    sum = addNumbers(n1, n2);

    ... .. ...
}

int addNumbers(int a, int b)
{
    ... .. ...
    ... .. ...
}
```

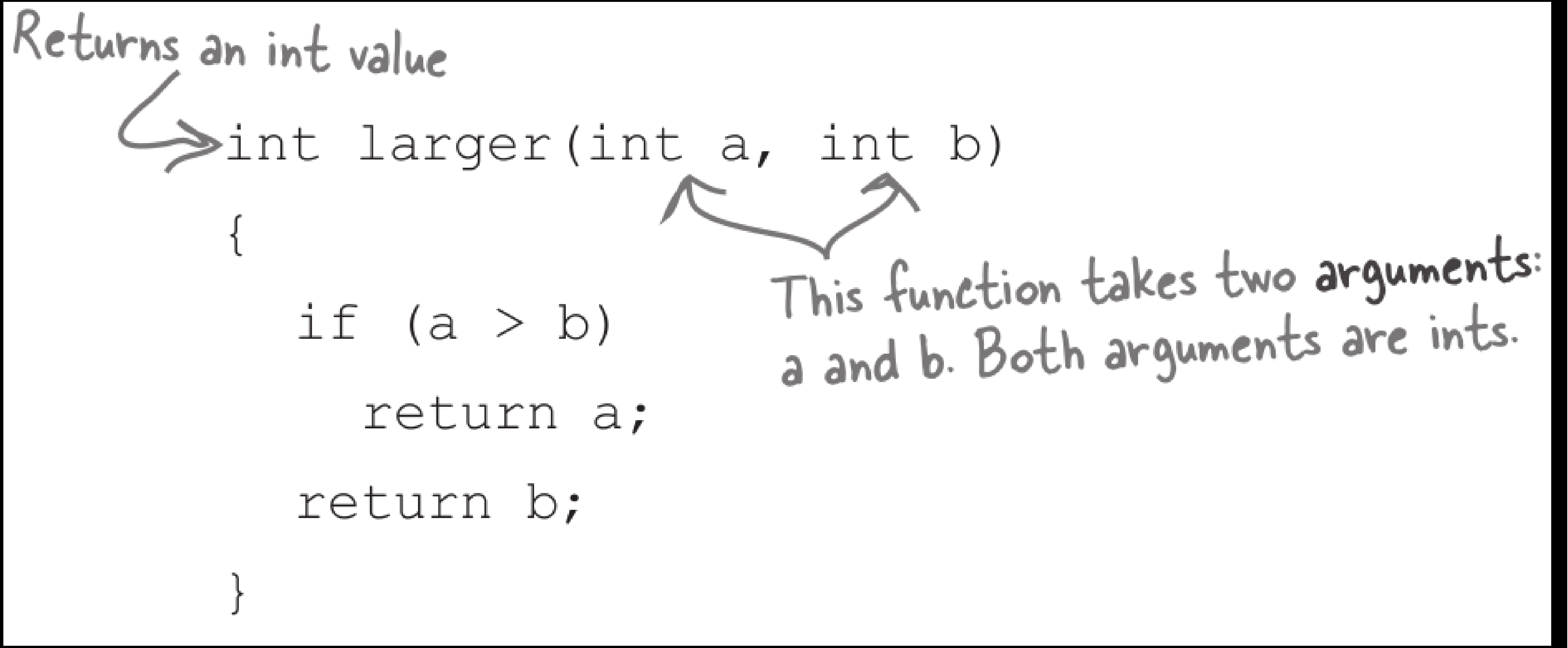
How are They Used?



```
#include <stdio.h>

int addNumbers(int a, int b);

int main()
{
    ... .. ...

    sum = addNumbers(n1, n2);

    ... .. ...
}

int addNumbers(int a, int b)
{
    ... .. ...
    return result;
}
```

Credits: programiz.com

Our `larger` Function



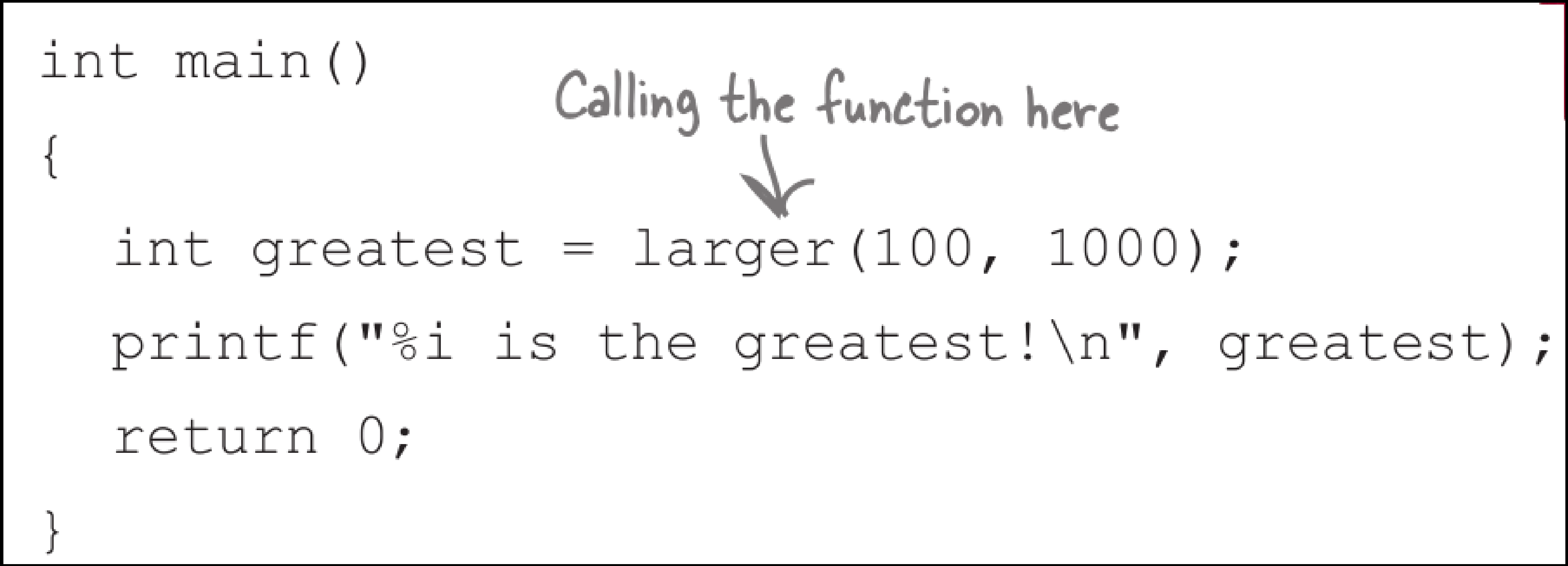
```
int larger(int a, int b)
{
  if (a > b)
    return a;
  return b;
}
```

Returns an int value
This function takes two arguments: a and b. Both arguments are ints.

Credits: Head First C

Is Called Here!!

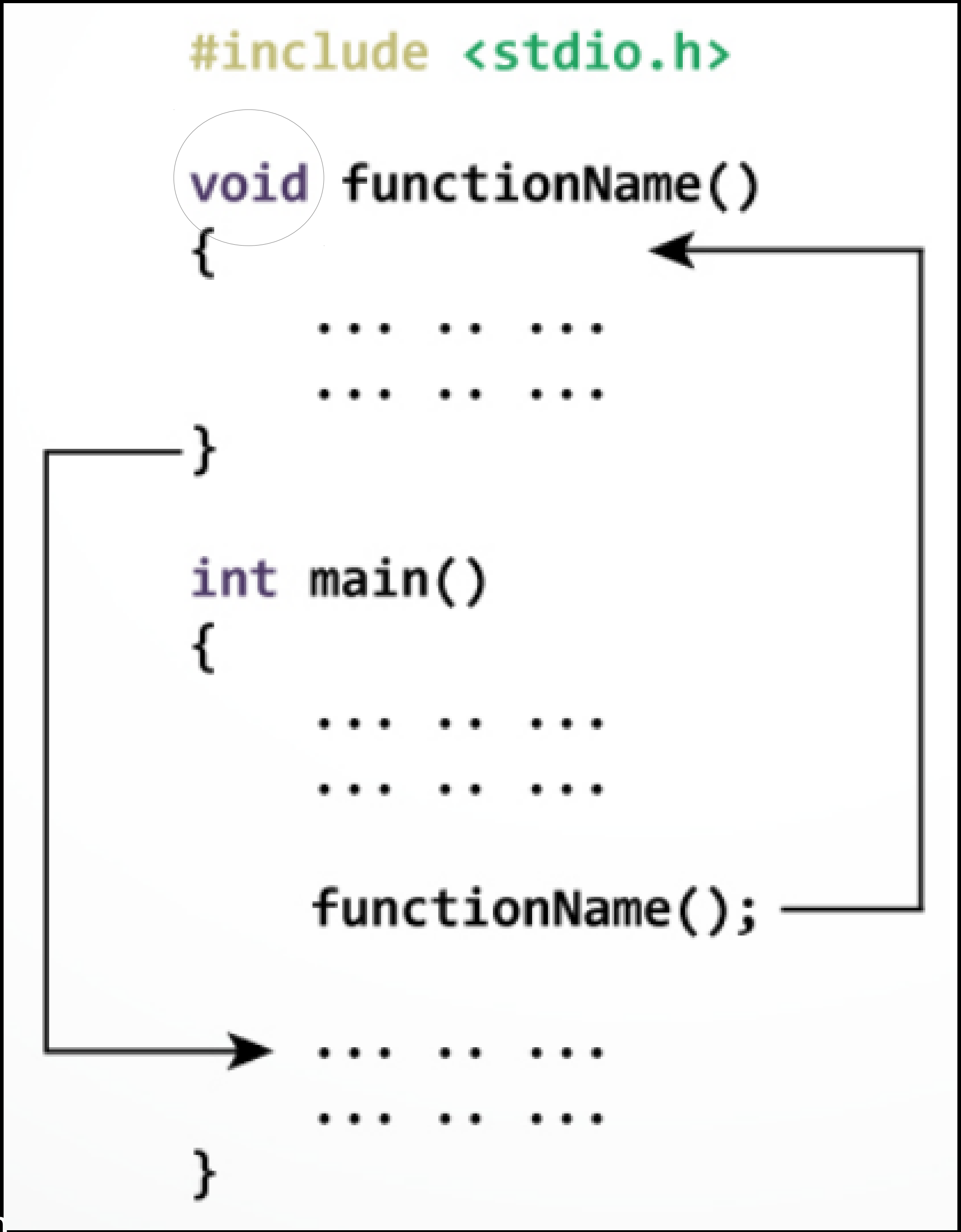
Assuming that function prototyping is done already



```
int main()
{
  int greatest = larger(100, 1000);
  printf("%i is the greatest!\n", greatest);
  return 0;
}
```

Credits: Head First C

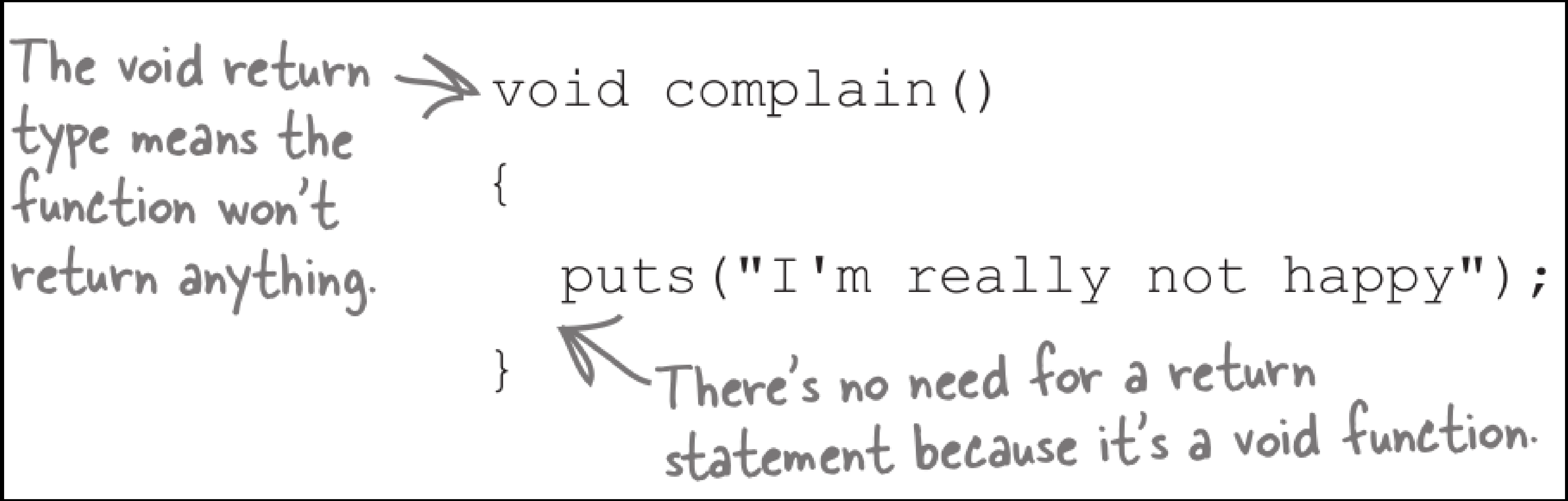
What's `void` in Function?



```
#include <stdio.h>

void functionName()
{
    ... .. ...
    ... .. ...
}

int main()
{
    ... .. ...
    ... .. ...

    functionName();

    ... .. ...
    ... .. ...
}
```

Credits: programiz.com

What's `void` in Function?



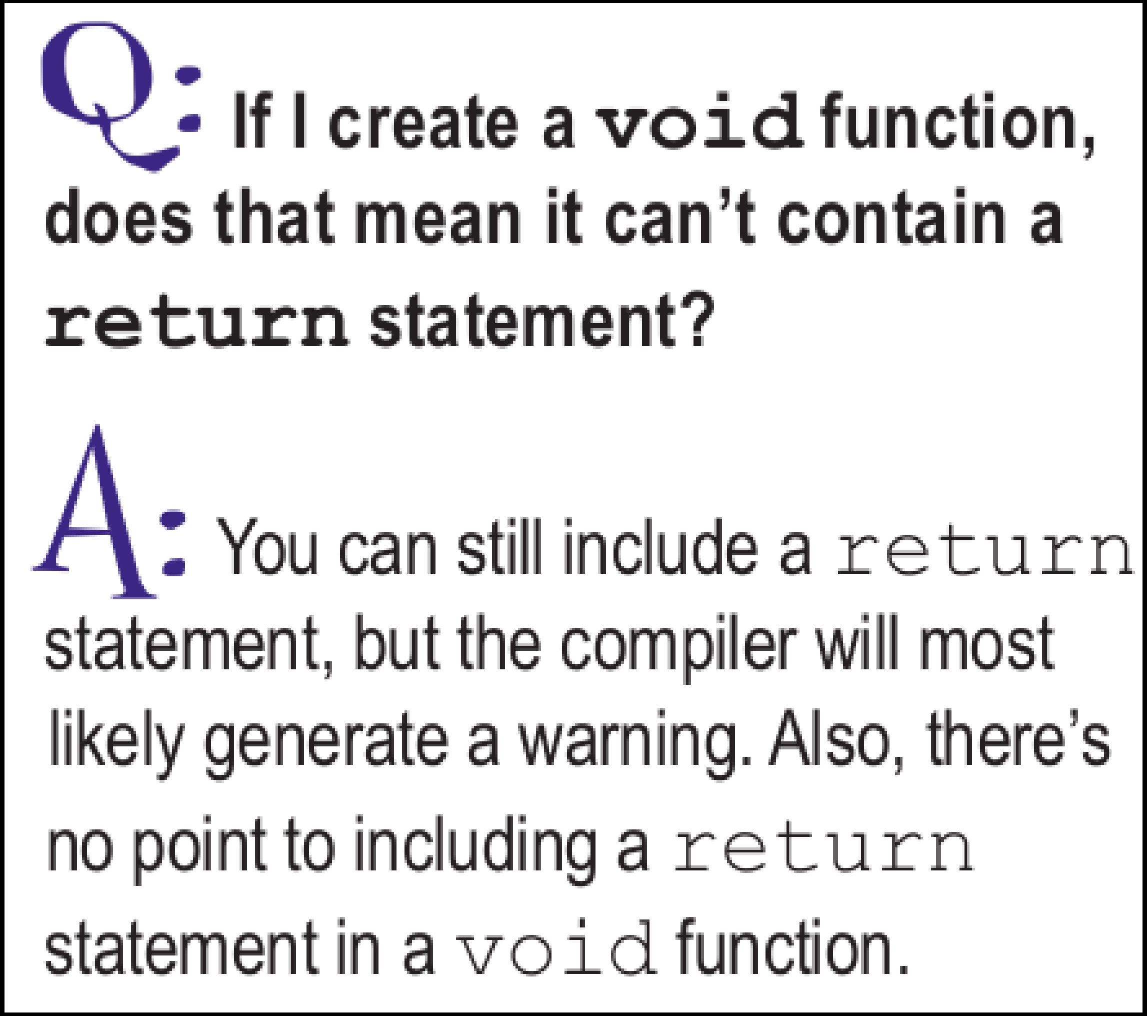
The void return type means the function won't return anything.

```
void complain()
{
   puts("I'm really not happy");
}
```

There's no need for a return statement because it's a void function.

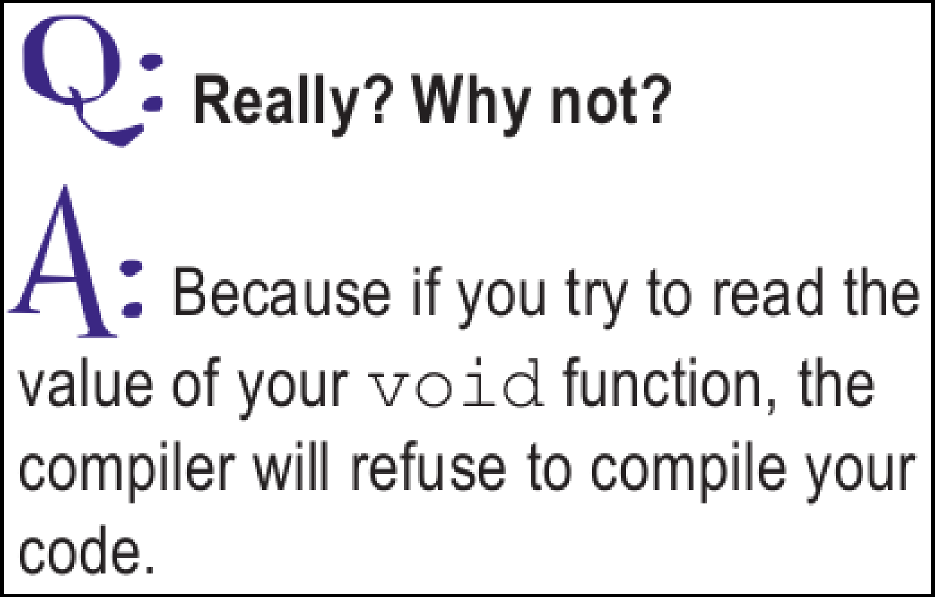
Credits: Head First C

there are no Dumb Questions



Q: If I create a `void` function, does that mean it can't contain a `return` statement?

A: You can still include a `return` statement, but the compiler will most likely generate a warning. Also, there's no point to including a `return` statement in a `void` function.



Q: Really? Why not?

A: Because if you try to read the value of your `void` function, the compiler will refuse to compile your code.

Credits: Head First C

Types of Functions

Standard library functions:

built-in functions

often defined in header files

available if header files are included

User-defined functions:

custom created based on requirements

Credits: programiz.com

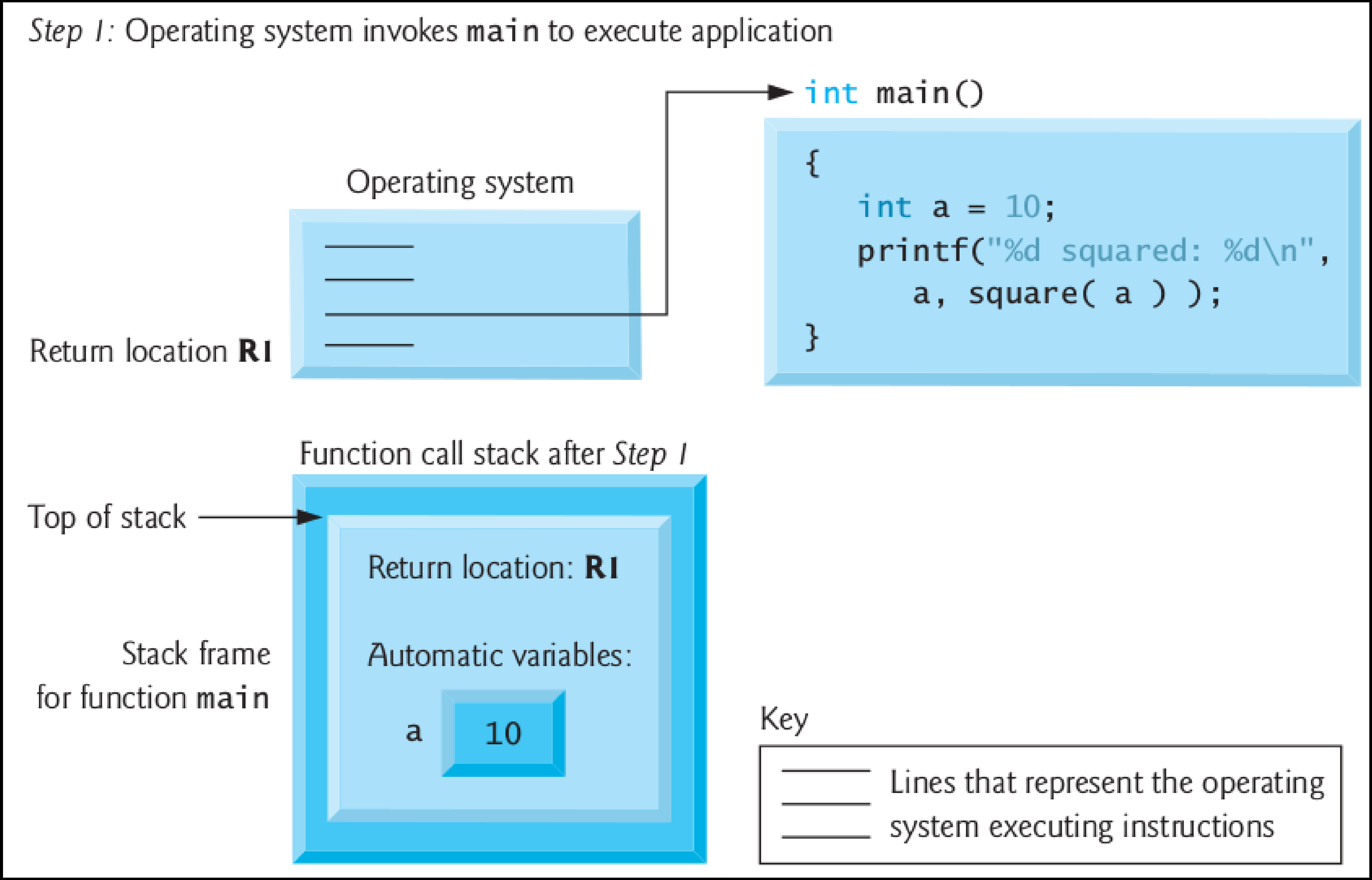
Types of User-Defined Functions

User-defined functions with

- no arguments and no return value
- no arguments and a return value
- arguments and no return value
- arguments and a return value

Credits: programiz.com

Under the Hood!!

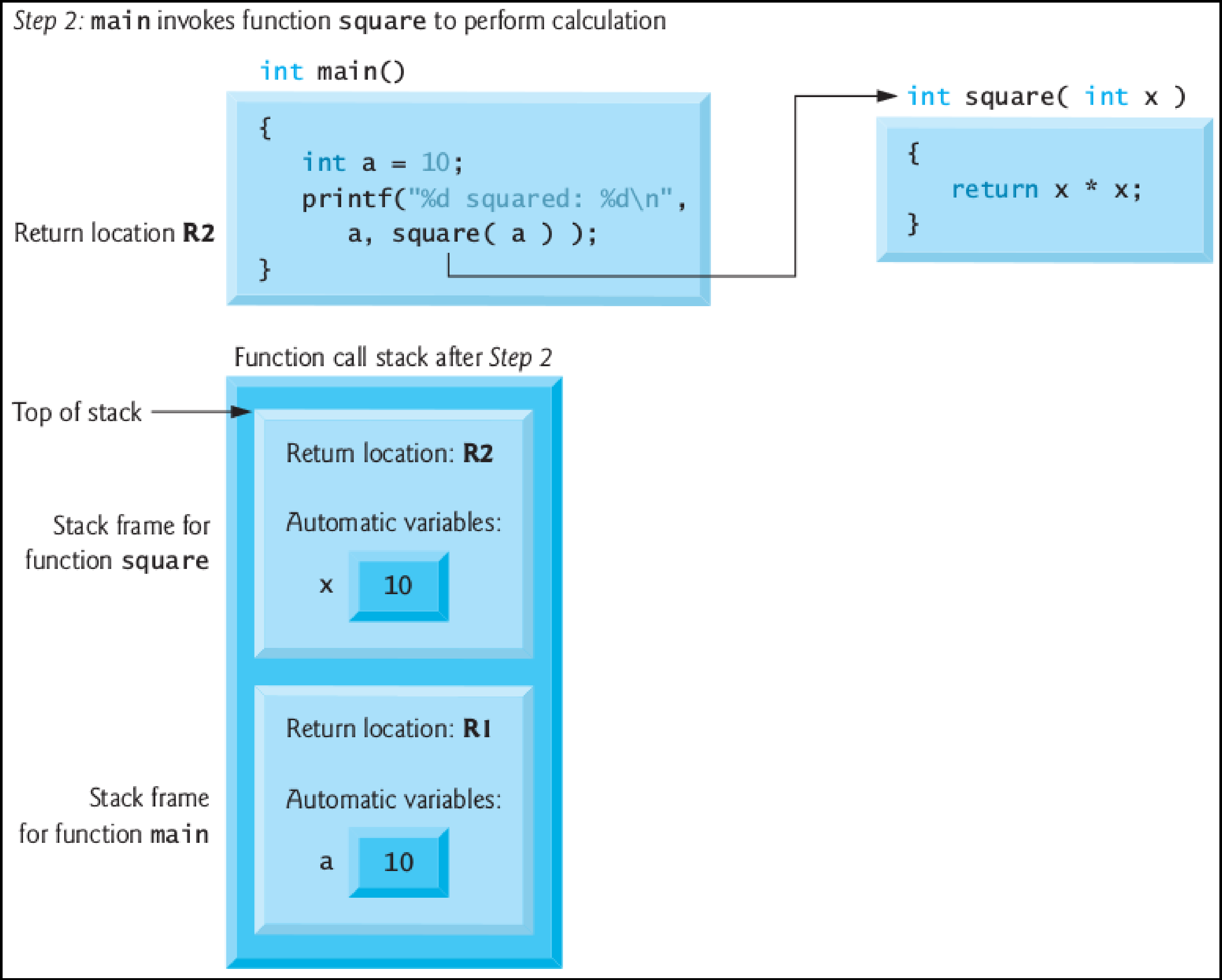


Step 1: Operating system invokes main to execute application
Operating system
Return location **R1**
int main()
Function call stack after *Step 1*
Top of stack
Return location: **R1**
Automatic variables:
a
10
Stack frame for function main
Key
Lines that represent the operating system executing instructions

```
int main()
{
   int a = 10;
   printf("%d squared: %d\n",
      a, square( a ) );
}
```

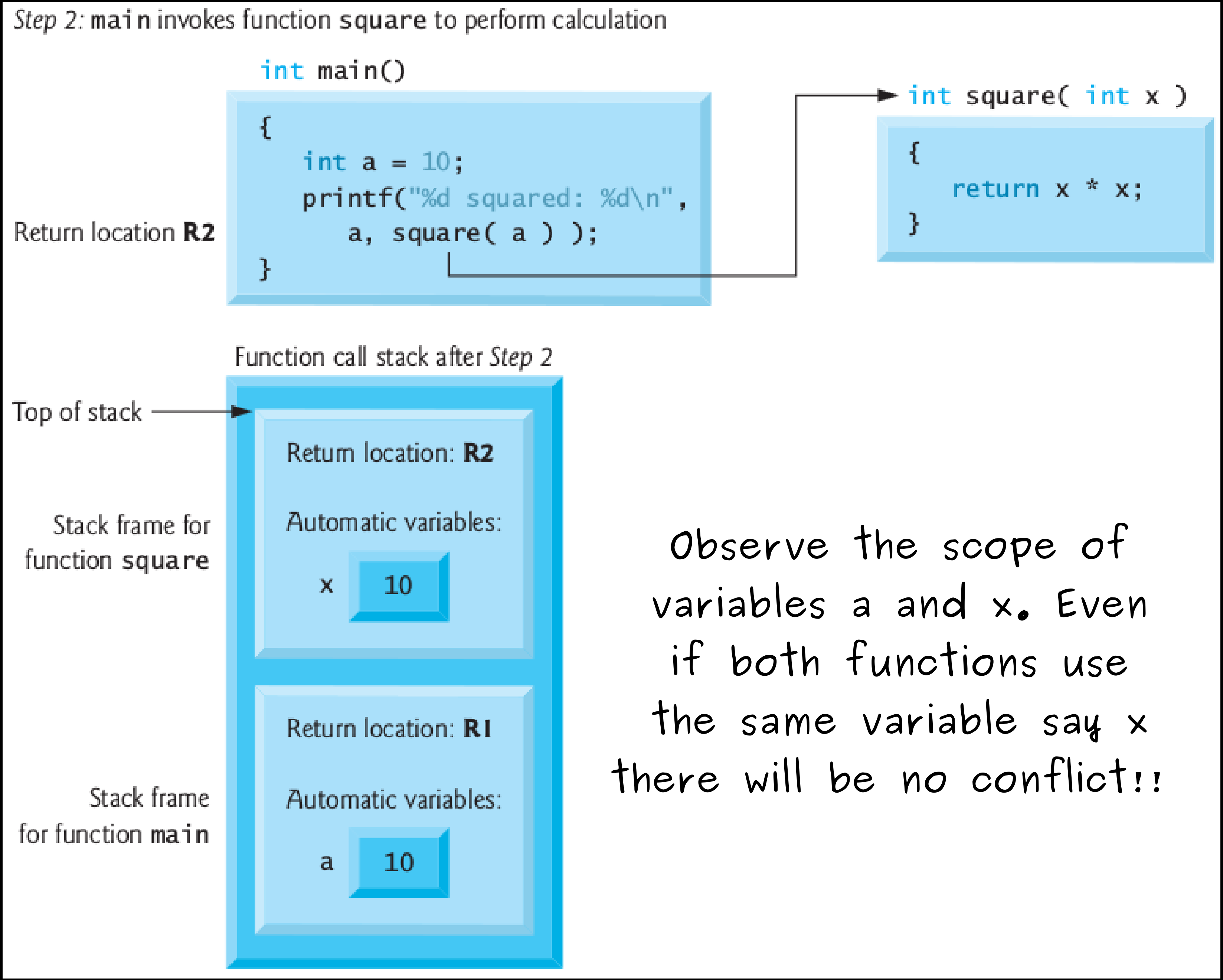
Credits: How to Program C

Under the Hood!!



Credits: How to Program C

Under the Hood!!



Step 2: main invokes function square to perform calculation
Return location R2
Function call stack after *Step 2*
Top of stack
Stack frame for function square
Return location: R2
Automatic variables:
x 10
Stack frame for function main
Return location: R1
Automatic variables:
a 10

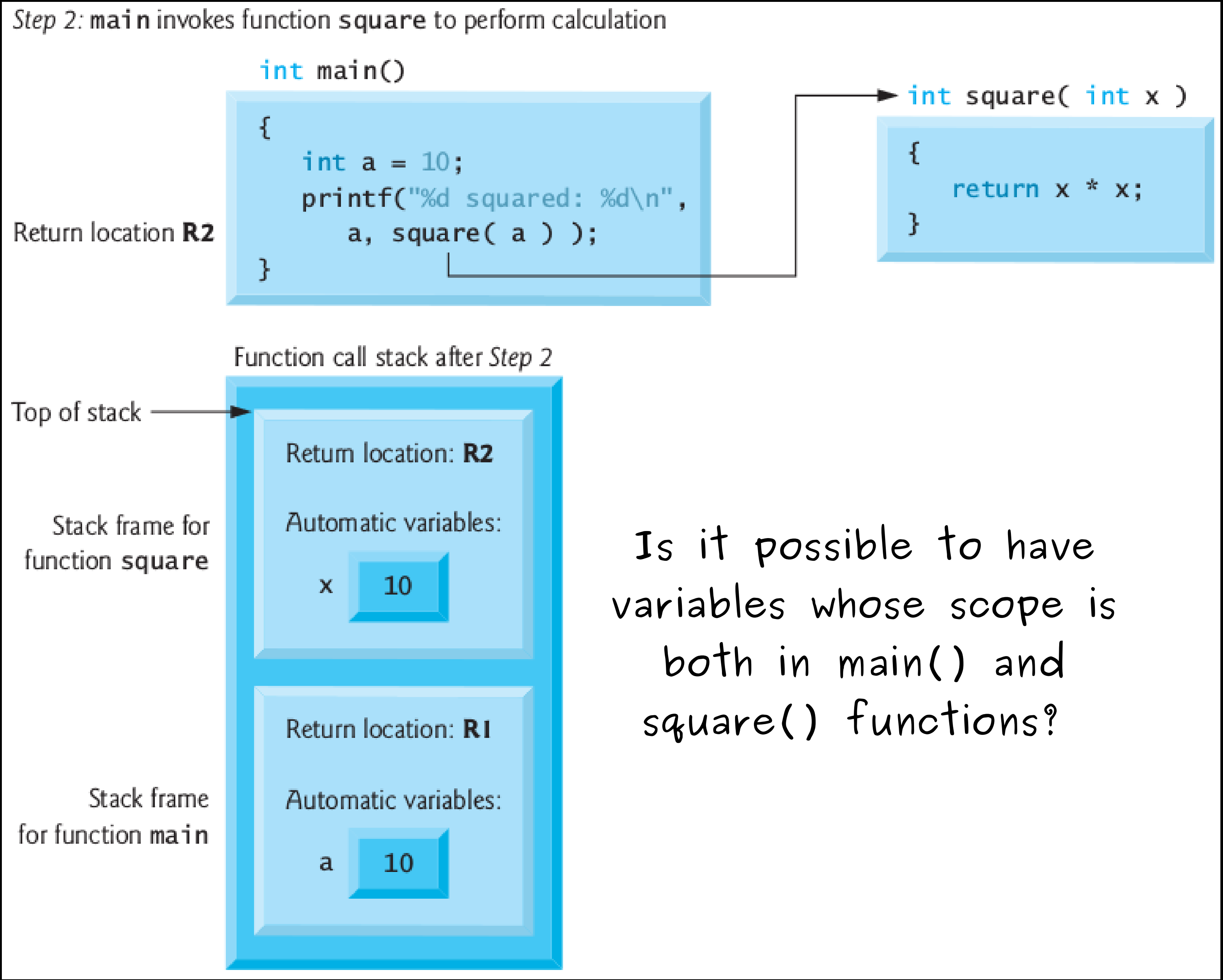
```
int main()
{
   int a = 10;
   printf("%d squared: %d\n",
      a, square( a ) );
}
```

```
int square( int x )
{
   return x * x;
}
```

Observe the scope of variables a and x. Even if both functions use the same variable say x there will be no conflict!!

Credits: How to Program C

Under the Hood!!



Step 2: `main` invokes function `square` to perform calculation

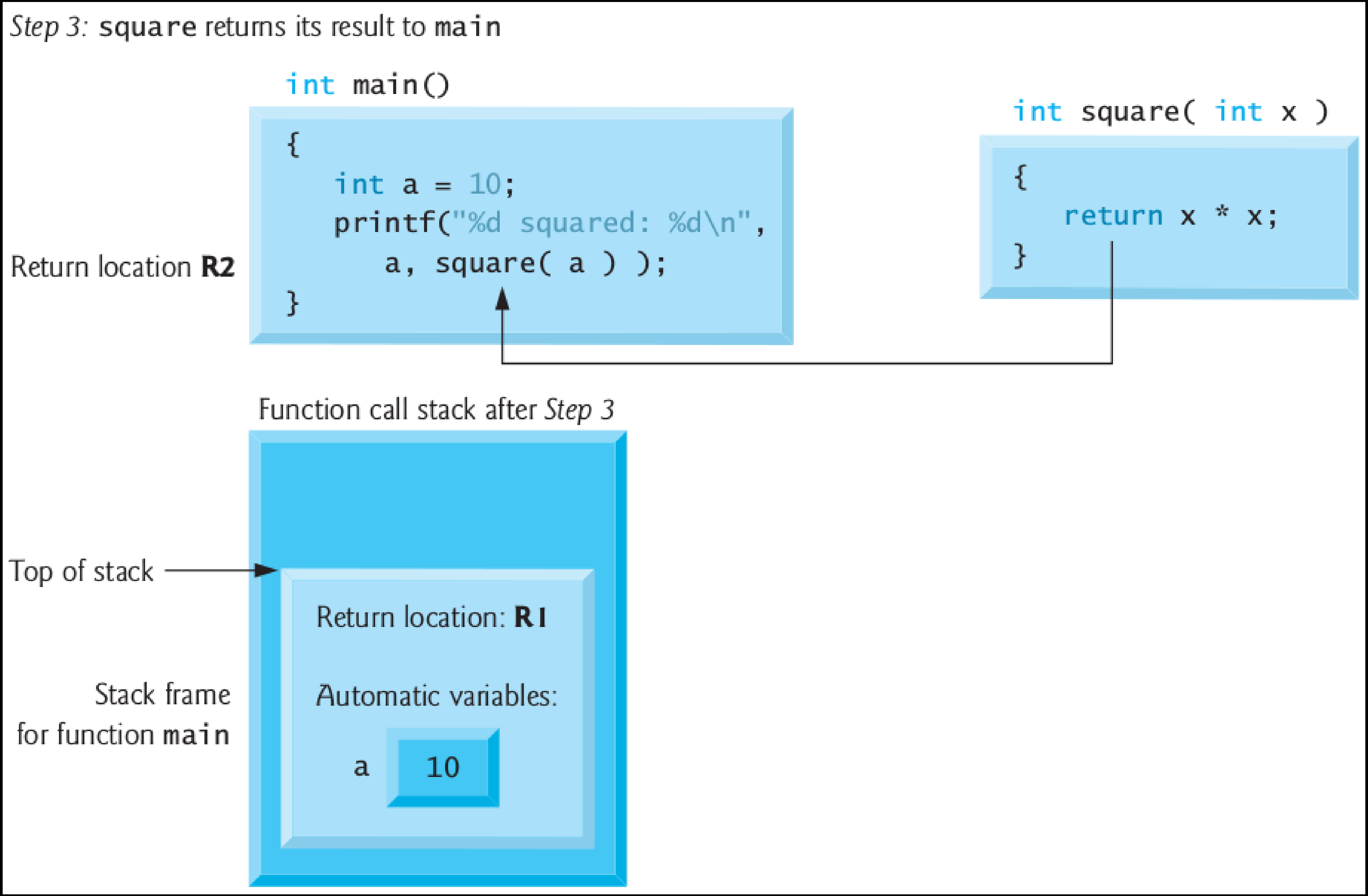
```c
int main()
{
    int a = 10;
    printf("%d squared: %d\n",
       a, square( a ) );
}
```

```c
int square( int x )
{
    return x * x;
}
```

Is it possible to have variables whose scope is both in main() and square() functions?

Credits: How to Program C

Under the Hood!!

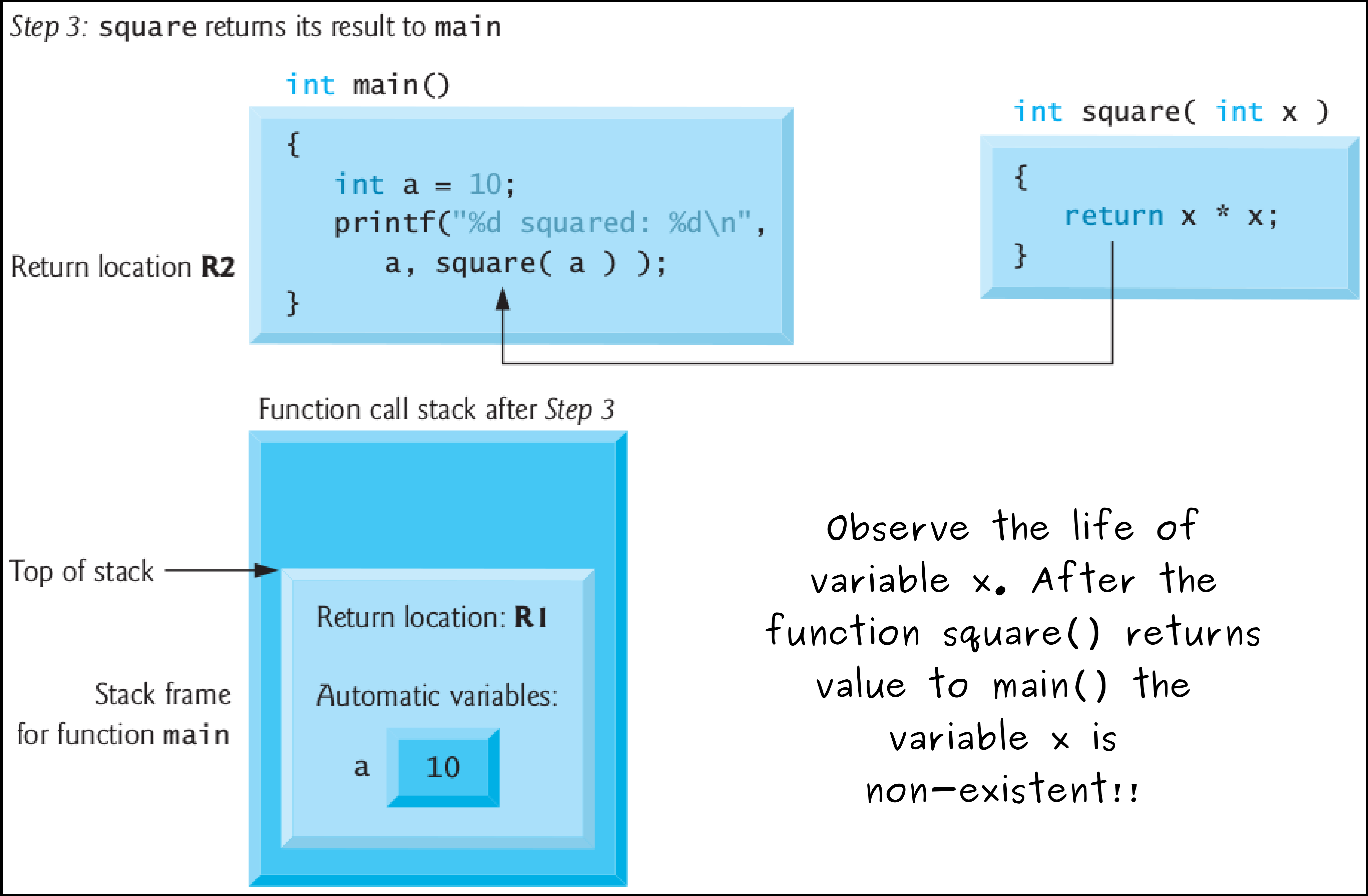

Step 3: `square` returns its result to `main`

```
int main()
{
   int a = 10;
   printf("%d squared: %d\n",
      a, square( a ) );
}
```

```
int square( int x )
{
   return x * x;
}
```

Credits: How to Program C

Under the Hood!!



Step 3: `square` returns its result to `main`

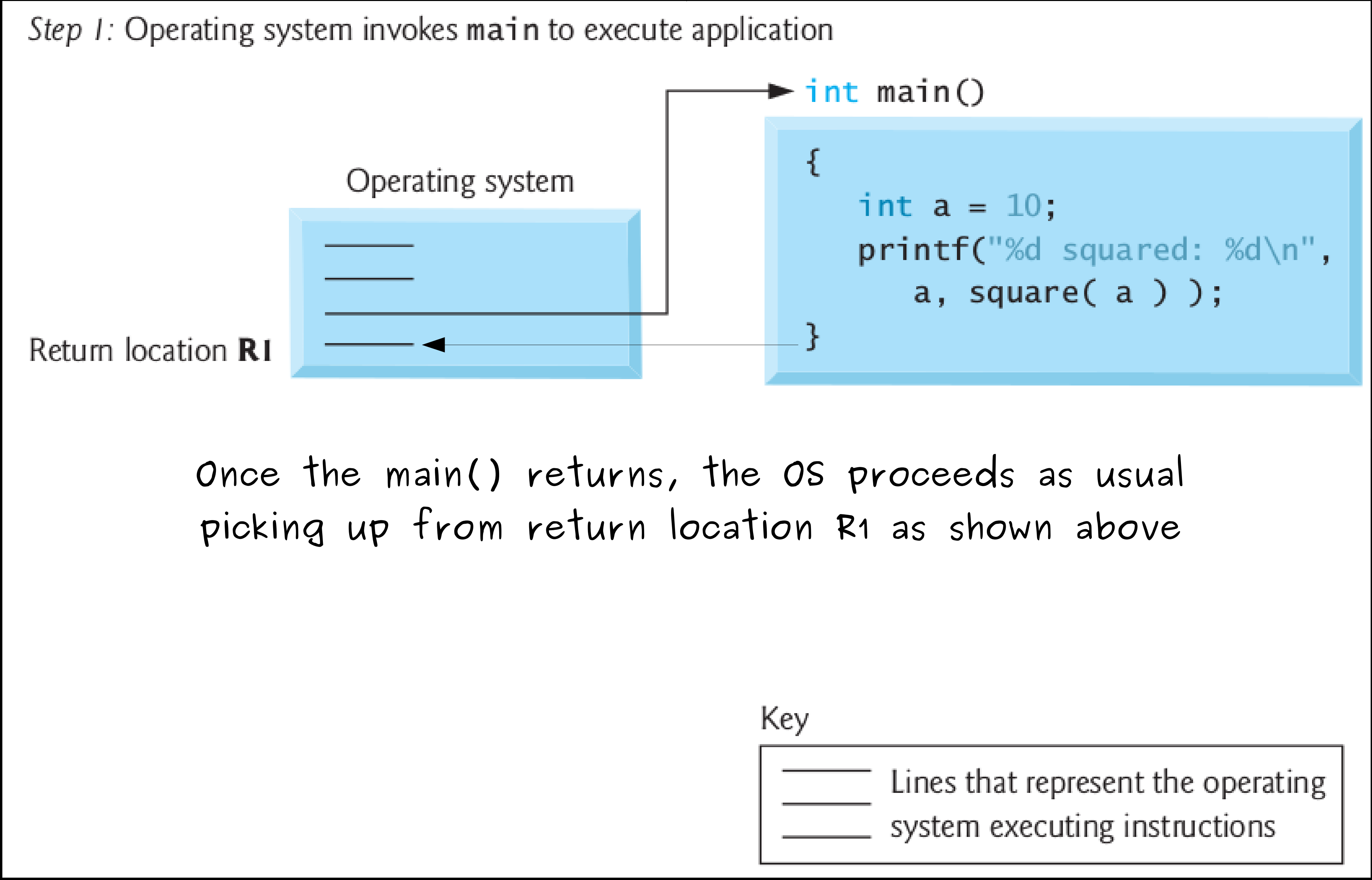
```
int main()
{
    int a = 10;
    printf("%d squared: %d\n",
        a, square( a ) );
}
```

```
int square( int x )
{
    return x * x;
}
```

Observe the life of variable x. After the function square() returns value to main() the variable x is non-existent!!

Credits: How to Program C

Under the Hood!!


Step 1: Operating system invokes main to execute application
Operating system
Return location **R1**
Key
Lines that represent the operating system executing instructions

```c
int main()
{
    int a = 10;
    printf("%d squared: %d\n",
        a, square( a ) );
}
```

Once the main() returns, the OS proceeds as usual picking up from return location R1 as shown above

Credits: How to Program C

CSE102
Computer Programming

(Next Topic)

Credits: psychologywizard.net, livingwithra-nan.blogspot.in