Ch1: Understand functions in C

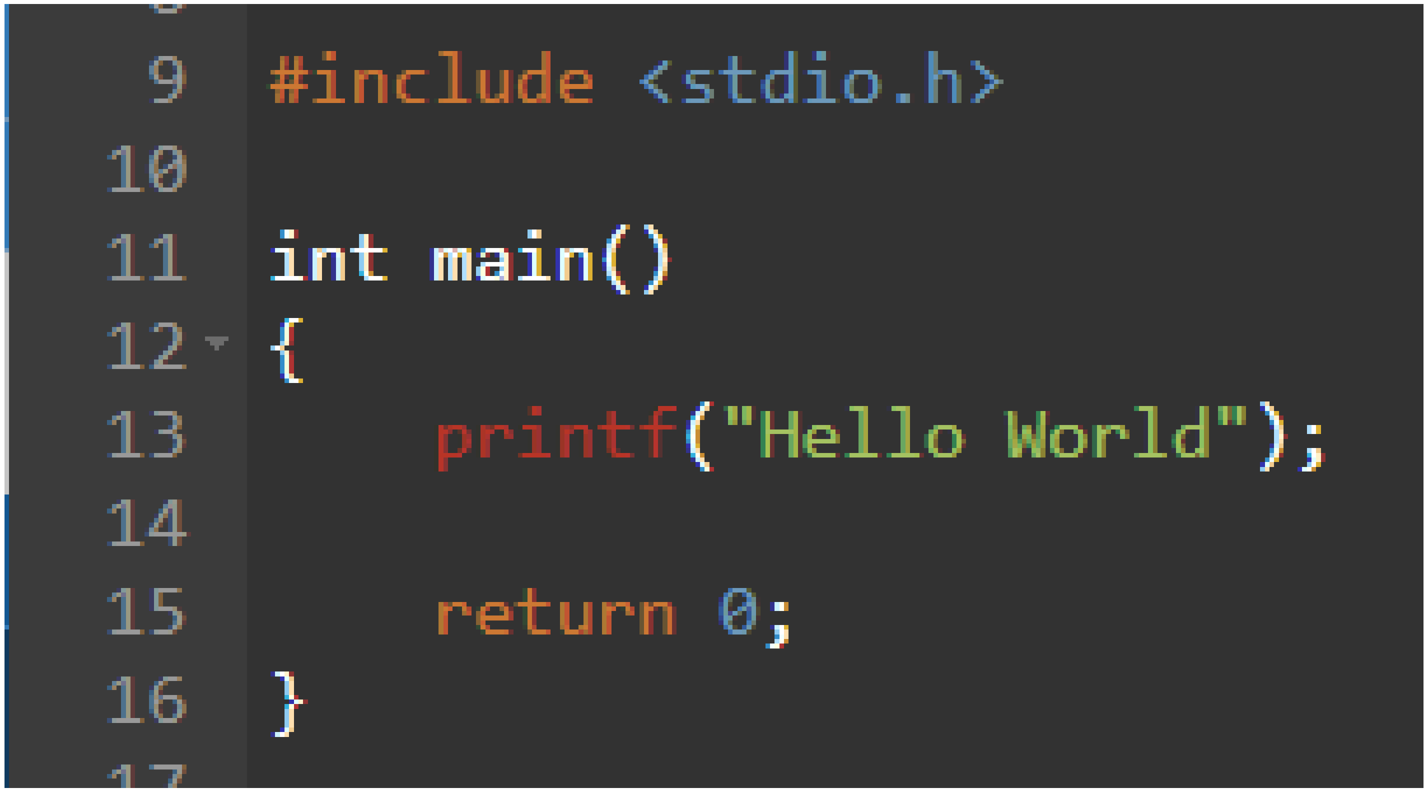
305171 Computer Programming

Jiraporn Pooksook

Naresuan University

How the compiler works

- It runs line by line in main function from 11 to 16.

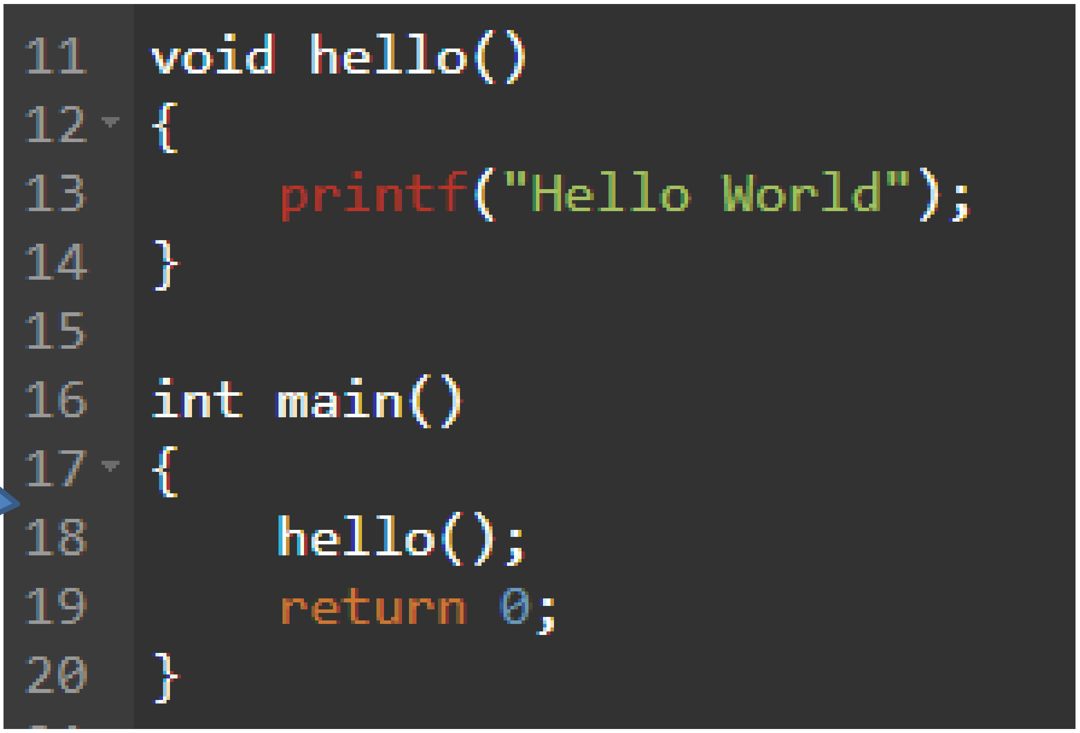


```
 9  #include <stdio.h>
10
11  int main()
12  {
13      printf("Hello World");
14
15      return 0;
16  }
```

How the compiler works

- It runs codes line by line in main function. When it sees a function, it jumps to finish the function and comes back to the main function again.

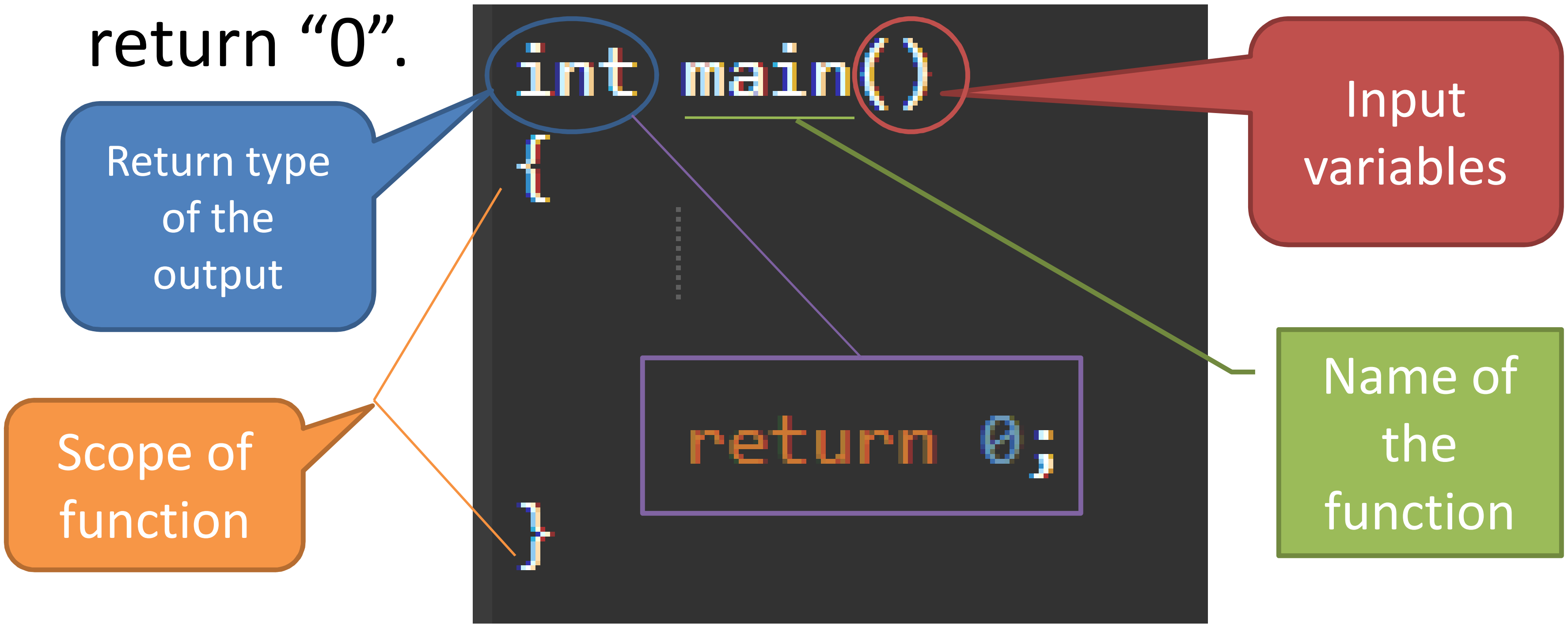
Run from 16-18, then jump to 11-14 and come back to 18-20



```
11  void hello()
12  {
13      printf("Hello World");
14  }
15
16  int main()
17  {
18      hello();
19      return 0;
20  }
```

The main function’s pattern

- The return type of output must be “int”.
- The name of function must be “main”.
- If the program ends succesfully , the function return “0”.



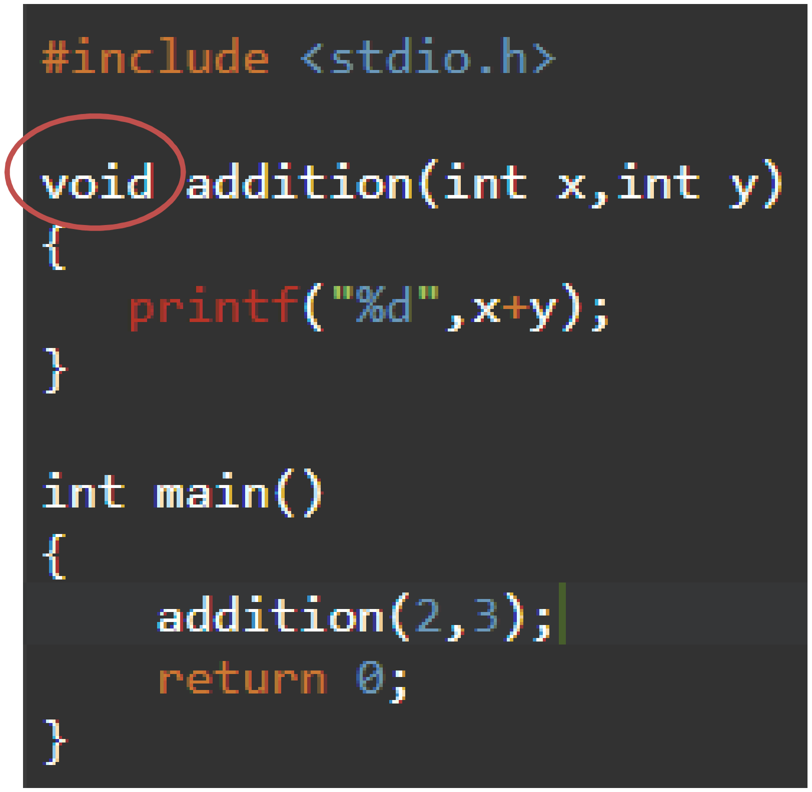
```
int main()
{

    return 0;
}
```

How to write a function

- There’re two types of function:
  - 1. Return some outputs
  - 2. Don’t return any output
- Function type 1:
  - The return type can be any types except “void”
  - Must have the keyword “return” at the last line of function.
- Function type 2:
  - The return type will be “void” only.

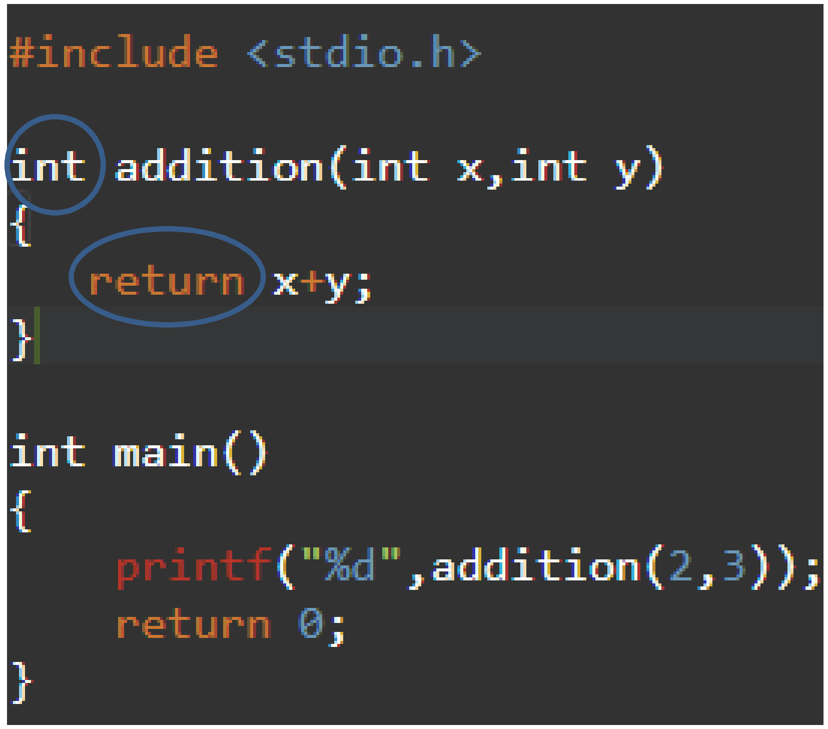
How to write a function

- Not return output



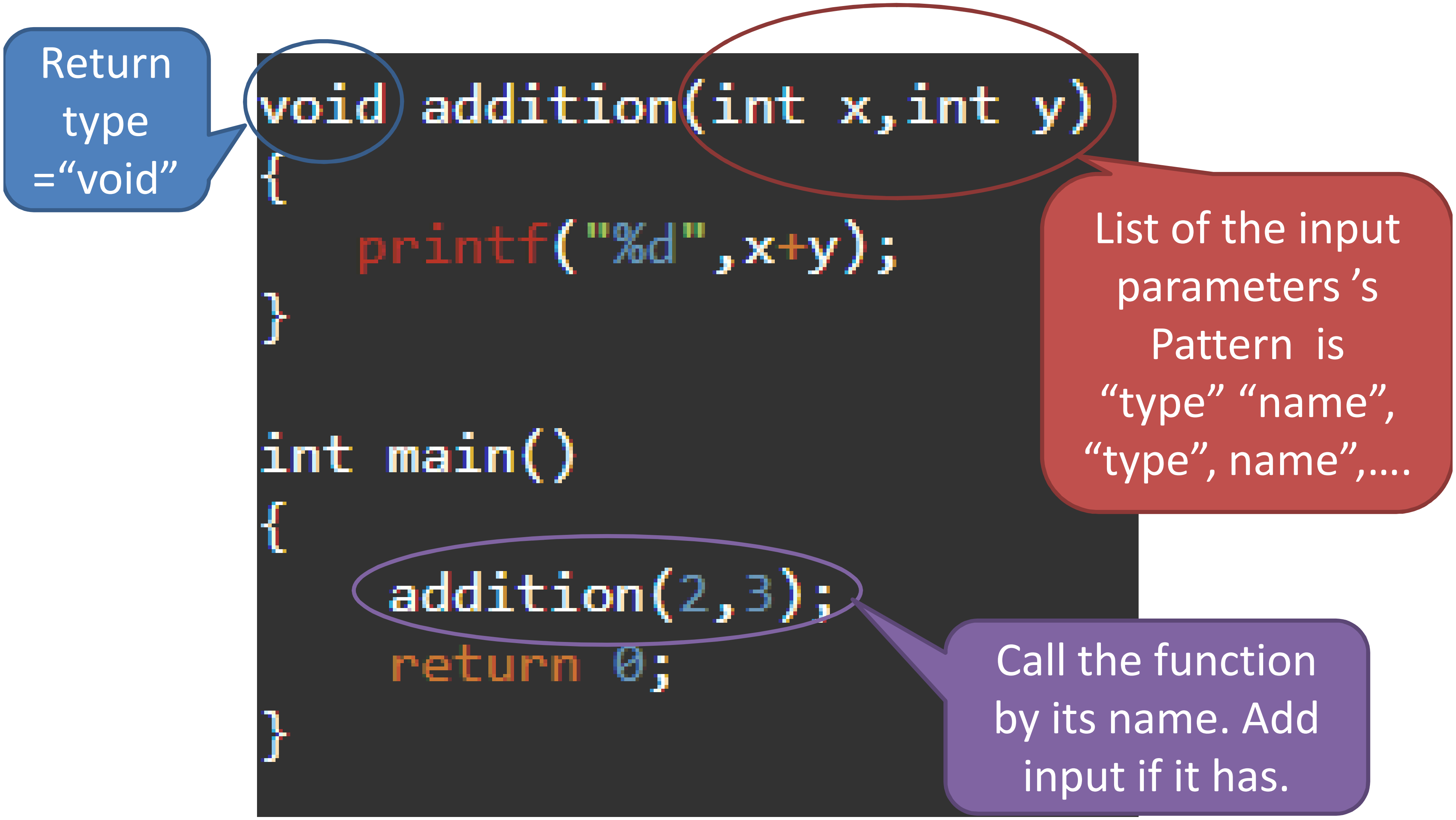
```c
#include <stdio.h>

void addition(int x,int y)
{
   printf("%d",x+y);
}

int main()
{
    addition(2,3);
    return 0;
}
```

- Return output



```c
#include <stdio.h>

int addition(int x,int y)
{
   return x+y;
}

int main()
{
    printf("%d",addition(2,3));
    return 0;
}
```

How to write a function: Not return

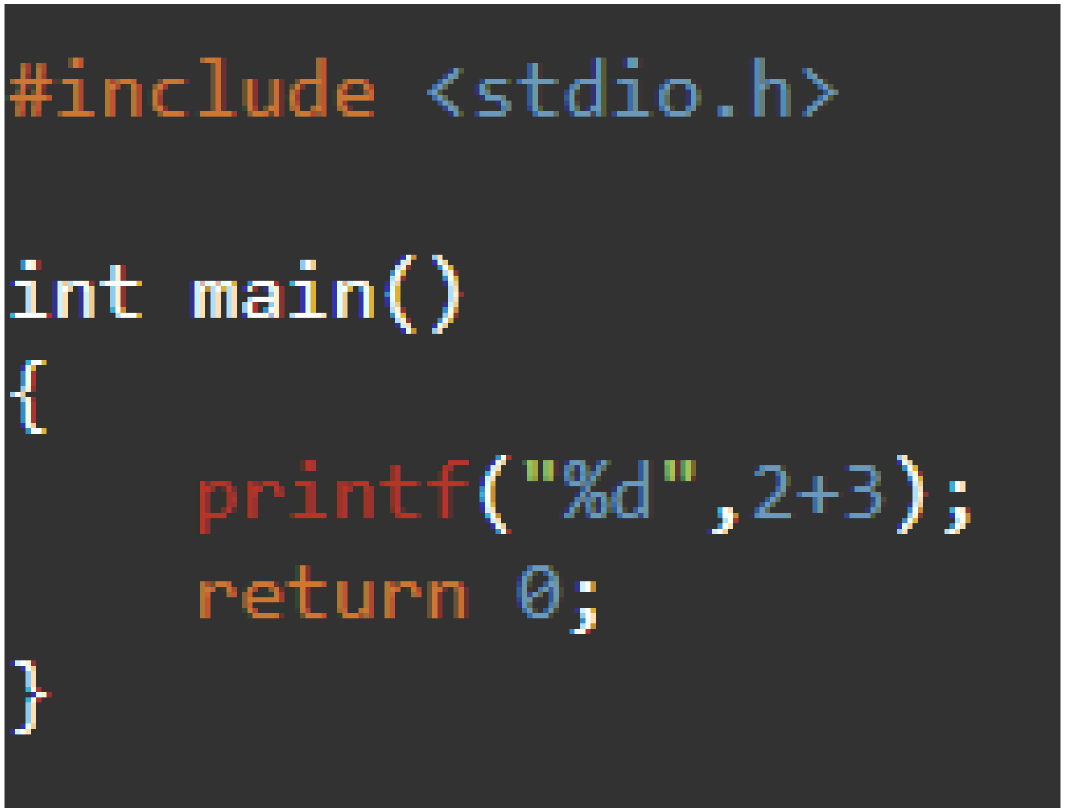


```
void addition(int x,int y)
{
    printf("%d",x+y);
}

int main()
{
    addition(2,3);
    return 0;
}
```

Return type ="void"
List of the input parameters 's Pattern is "type" "name", "type", name",….
Call the function by its name. Add input if it has.

Exercise: Understand functions

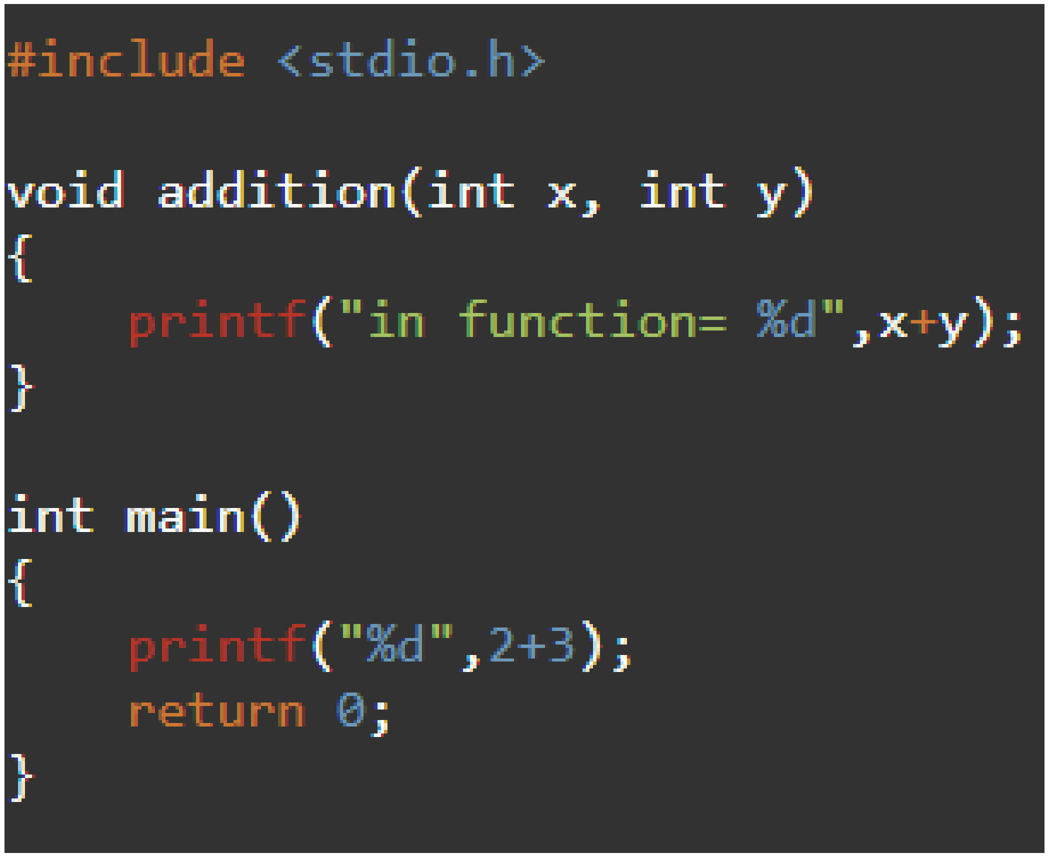
- There’s only the main function.



```c
#include <stdio.h>

int main()
{
    printf("%d",2+3);
    return 0;
}
```

Exercise: Understand functions

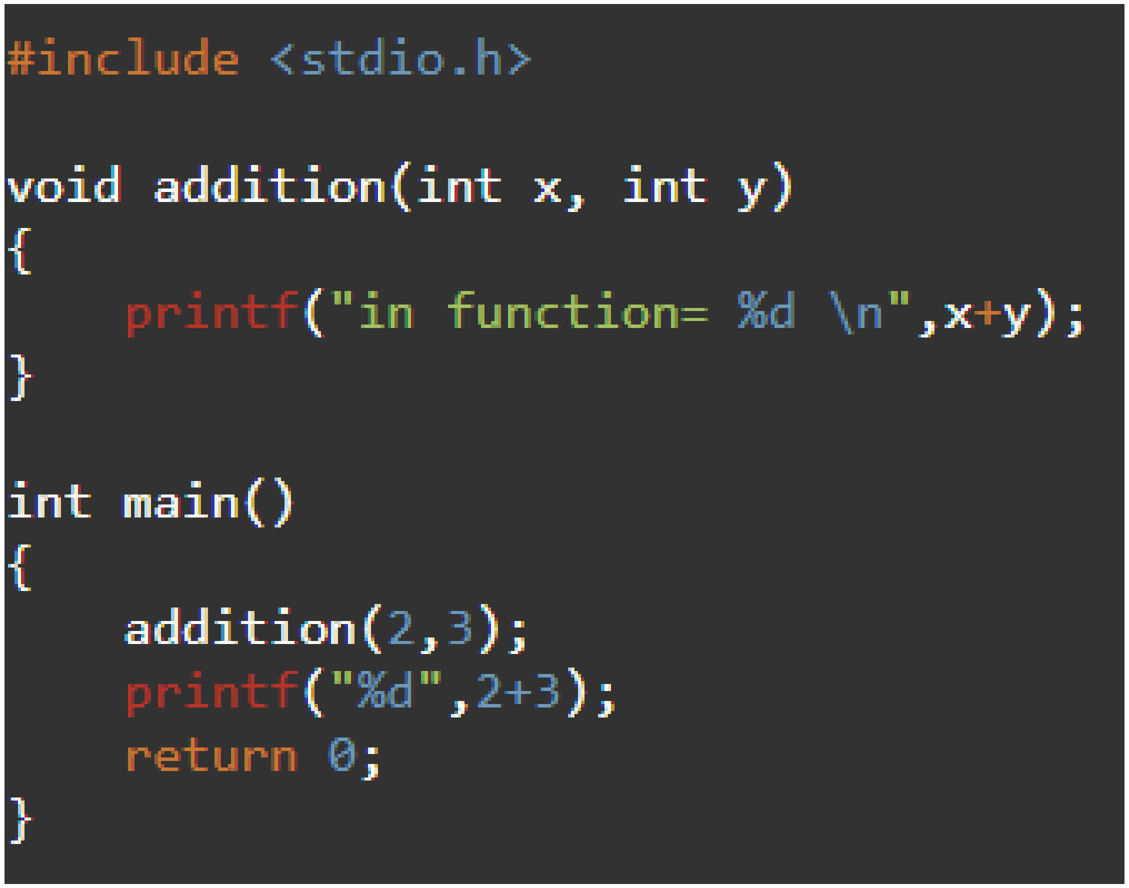
- Create a function “addition” but we don’t call it in the main function.



```c
#include <stdio.h>

void addition(int x, int y)
{
    printf("in function= %d",x+y);
}

int main()
{
    printf("%d",2+3);
    return 0;
}
```

Exercise: Understand functions

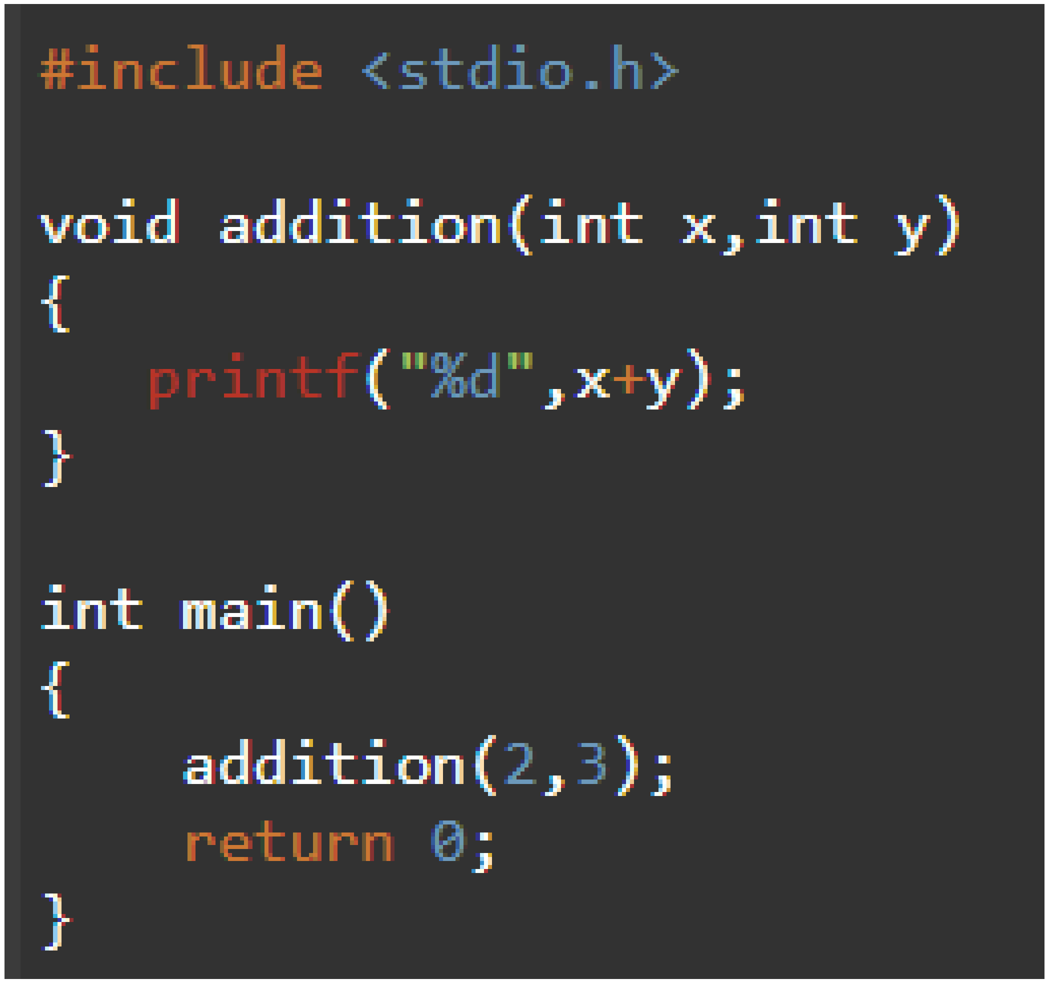
- Now we call the function “addition” in main.



```c
#include <stdio.h>

void addition(int x, int y)
{
    printf("in function= %d \n",x+y);
}

int main()
{
    addition(2,3);
    printf("%d",2+3);
    return 0;
}
```

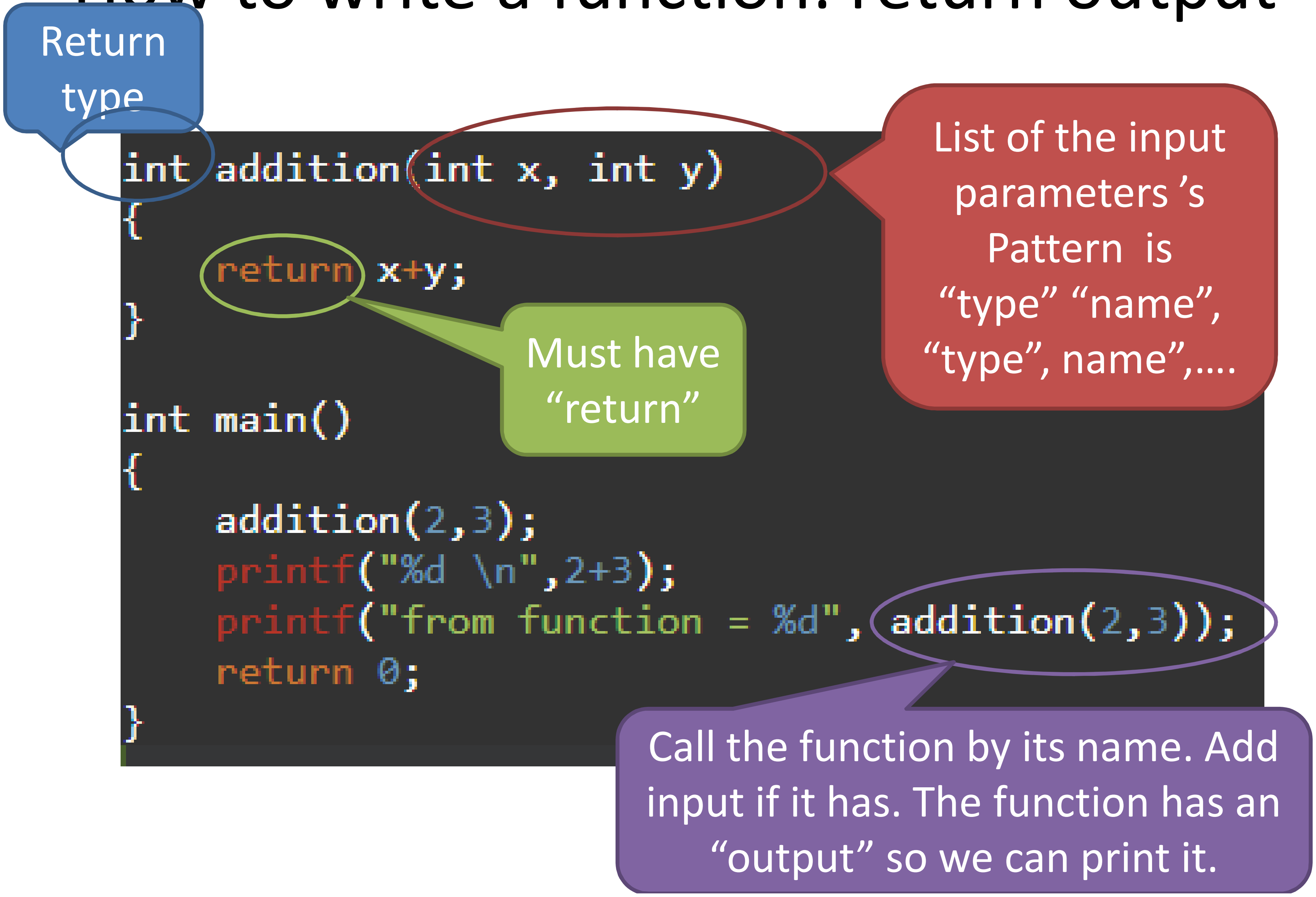
Exercise: Understand functions

- This is the perfect function we want.



```c
#include <stdio.h>

void addition(int x,int y)
{
   printf("%d",x+y);
}

int main()
{
    addition(2,3);
    return 0;
}
```

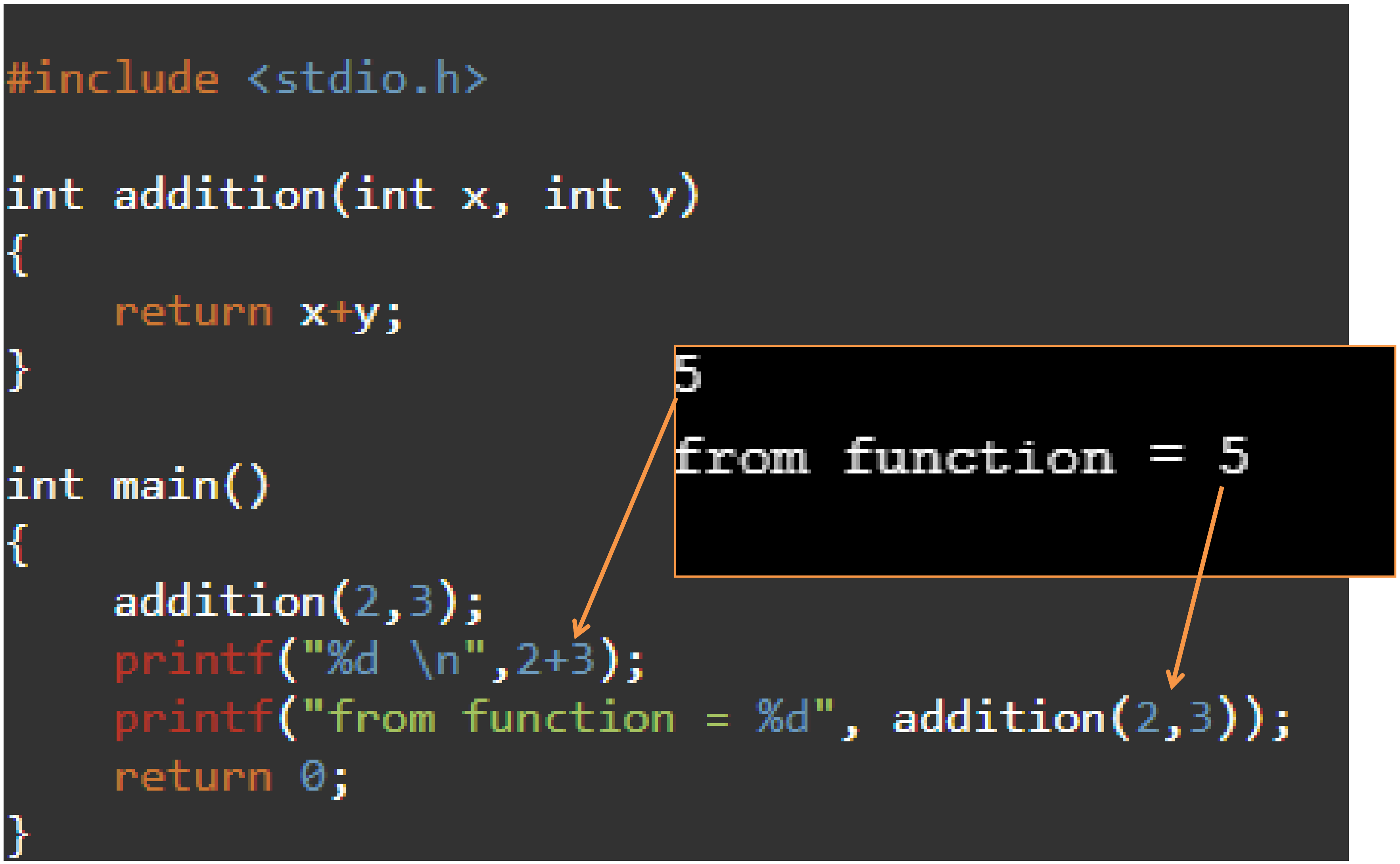
How to write a function: return output



```
int addition(int x, int y)
{
    return x+y;
}

int main()
{
    addition(2,3);
    printf("%d \n",2+3);
    printf("from function = %d", addition(2,3));
    return 0;
}
```

Return type
List of the input parameters 's Pattern is "type" "name", "type", name",….
Must have "return"
Call the function by its name. Add input if it has. The function has an "output" so we can print it.

Exercise: functions with output



```c
#include <stdio.h>

int addition(int x, int y)
{
    return x+y;
}

int main()
{
    addition(2,3);
    printf("%d \n",2+3);
    printf("from function = %d", addition(2,3));
    return 0;
}
```

```
5
from function = 5
```