Functions

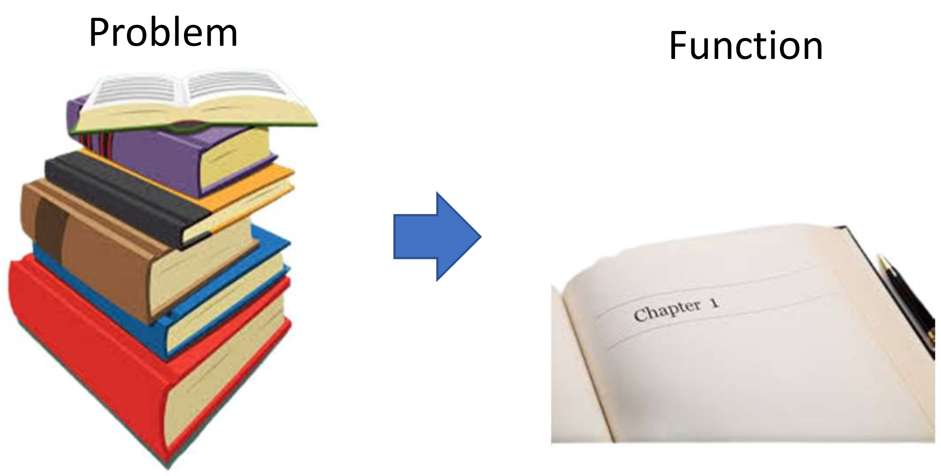
MMJ12503 – Computer programming

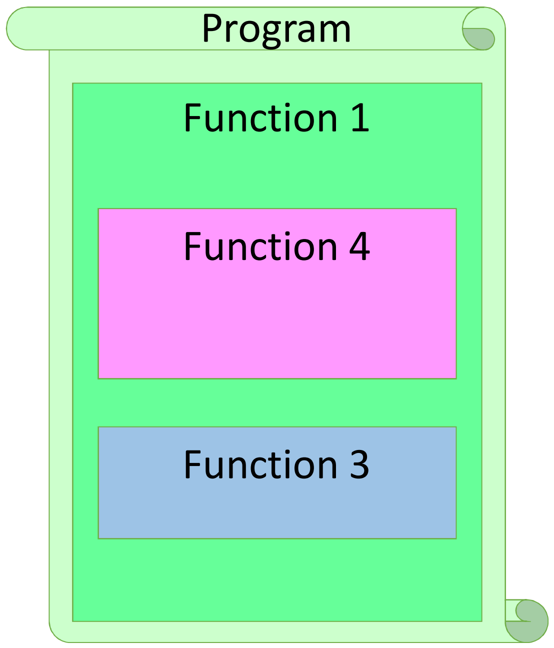
function()

Contents

Introduction

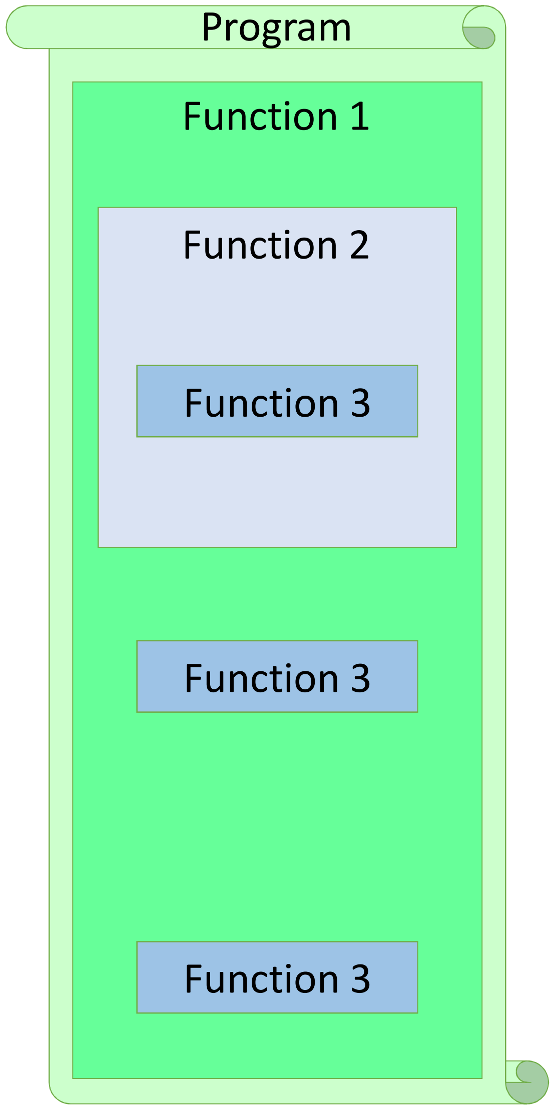
- A problem can be solved easily if it is decomposed into parts. Similarly a C program decomposes a program into its component functions.
- Big problems require big programs – too big to be written all at one time or to be written by a single programmer. Thus by decomposing a program into functions, we divide the work among several programmers.
- We can test the components of programs separately. We can change one function without changing or affecting the other functions.



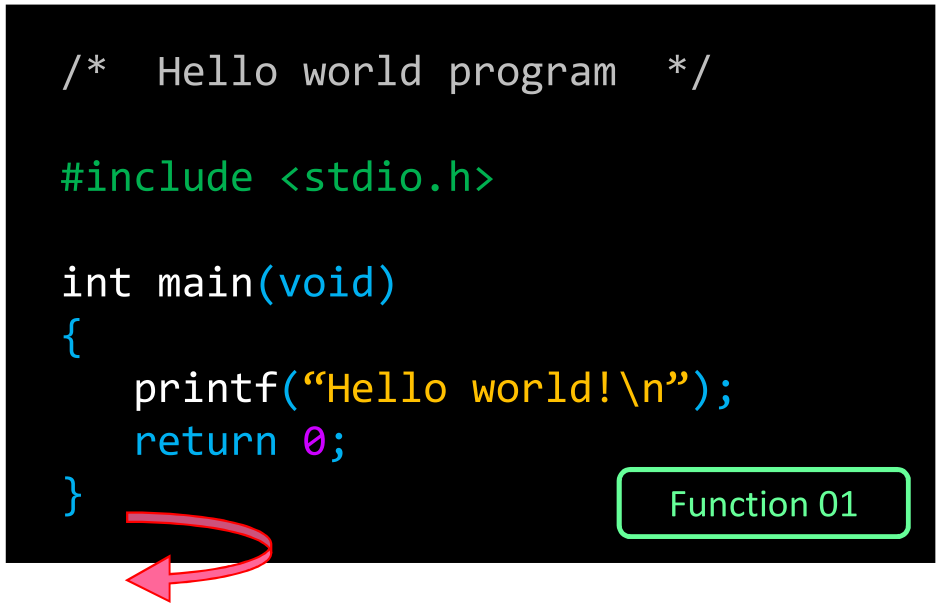


What is a function?

- Function is a series of statements that have been grouped together and given a name.
- In C, a function doesn’t necessarily have arguments, not does it necessarily compute a value.
- Functions provide a way to reuse code that is required in more than one place in your program.
- Function is an independent module and each function solves a part of the problem.



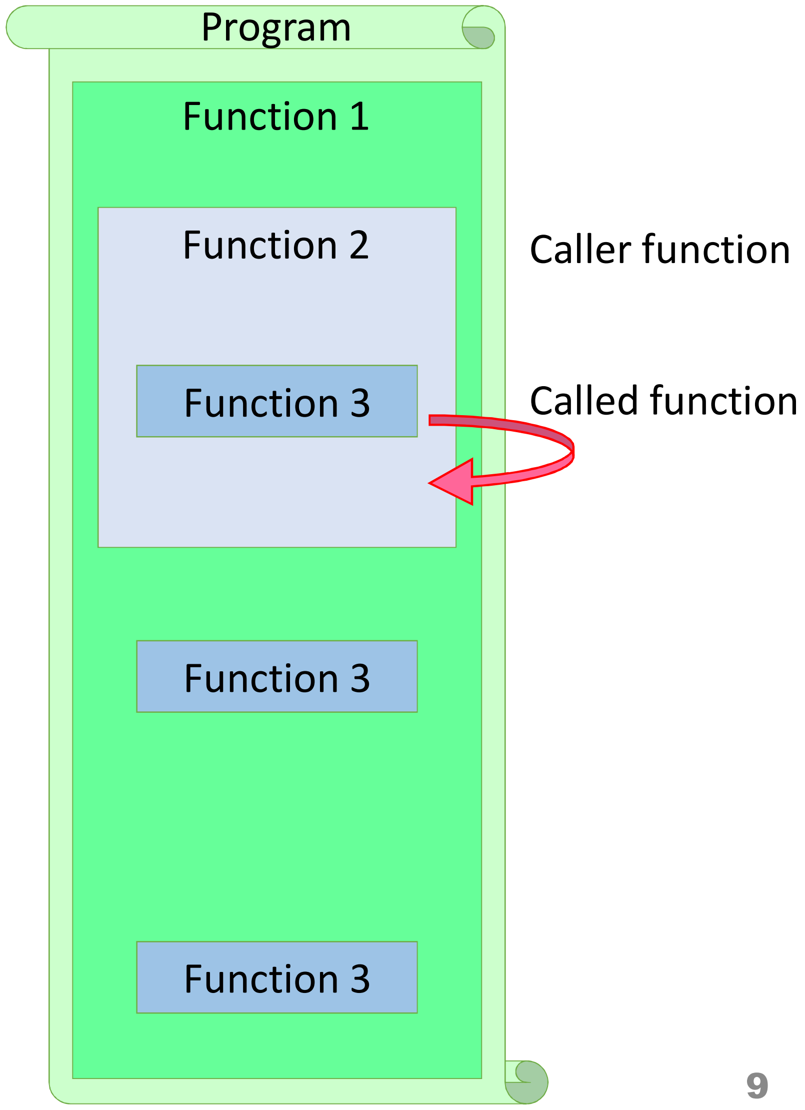
- Once after completing the task, the called function returns the control to the caller function.
- When main() completes its operations, control returns to the OS.

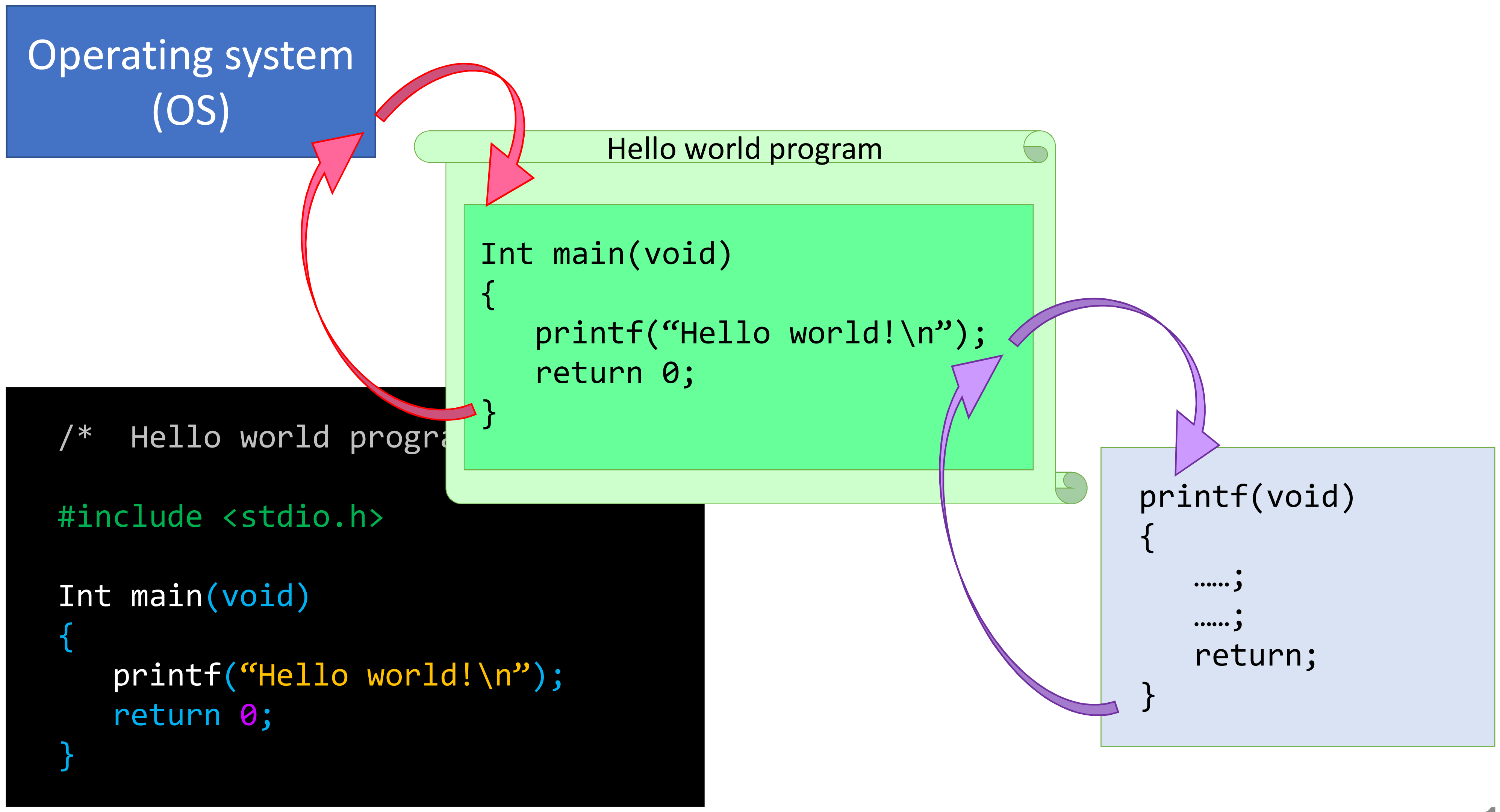


```
/*  Hello world program  */

#include <stdio.h>

int main(void)
{
   printf(“Hello world!\n”);
   return 0;
}
```

Main ()

OS



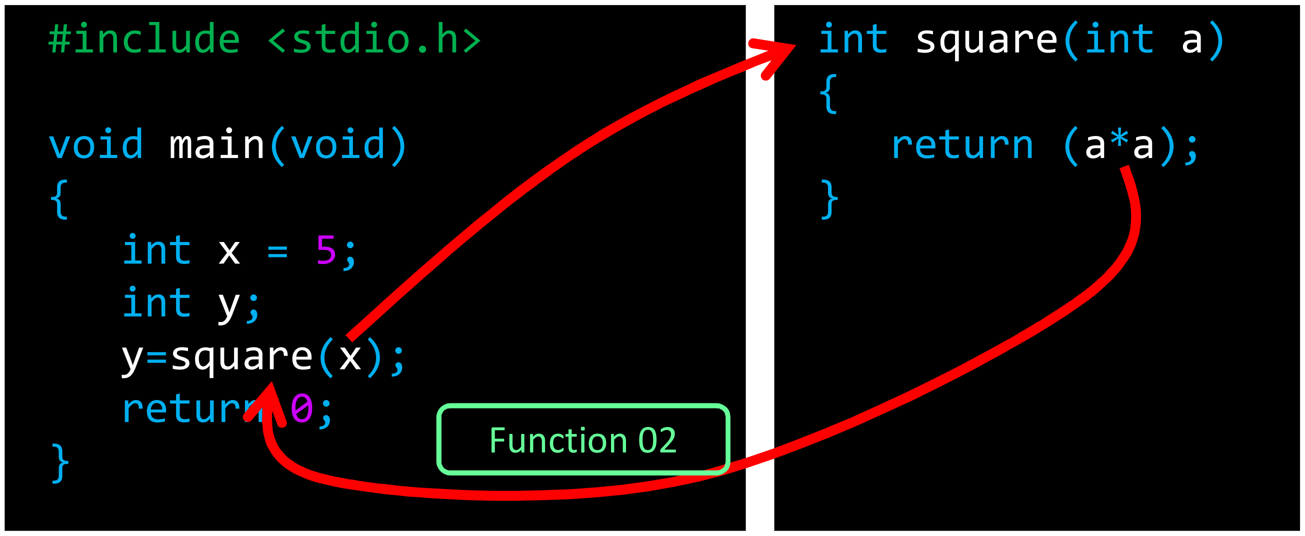


```
Int main(void)
{
    printf(“Hello world!\n”);
    return 0;
}
```

```
printf(void)
{
    ……;
    ……;
    return;
}
```

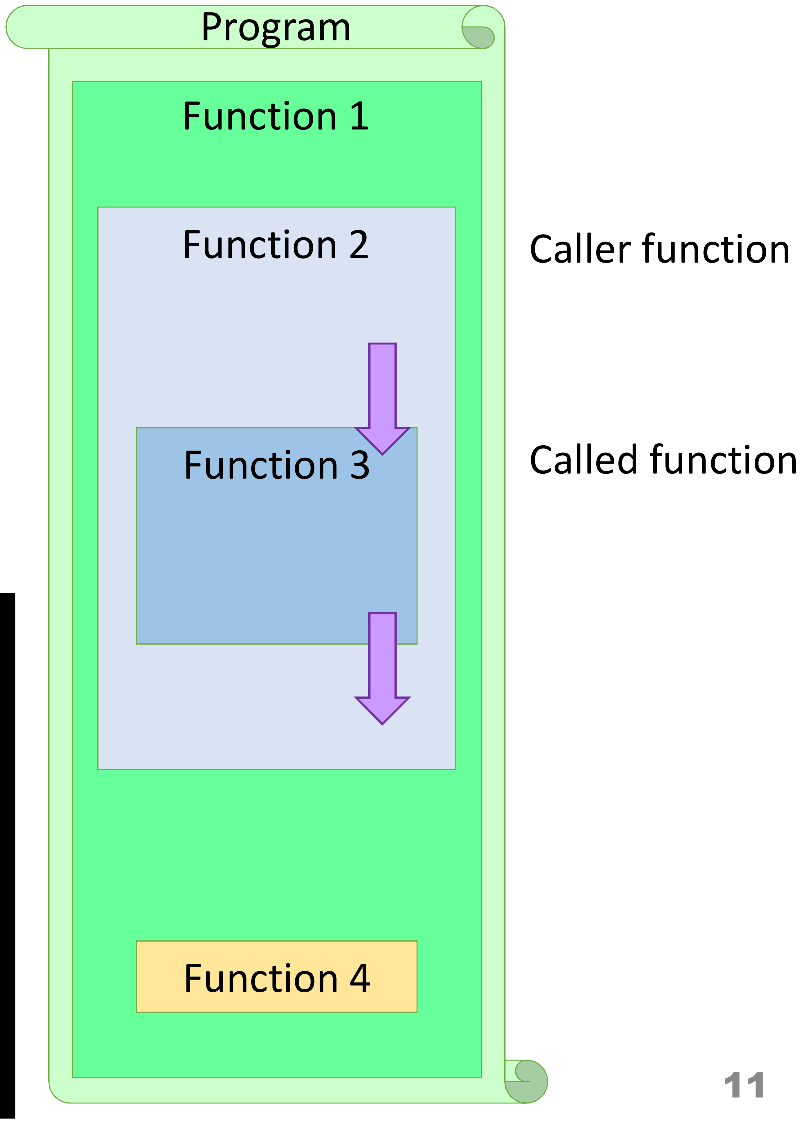
```
/*  Hello world progra

#include <stdio.h>

Int main(void)
{
    printf(“Hello world!\n”);
    return 0;
}
```

- In C, function can be invoked or called by another function.
- The caller function passes information to the called function.
- The called function may return information to the caller function.



```
#include <stdio.h>

void main(void)
{
    int x = 5;
    int y;
    y=square(x);
    return 0;
}
```

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



- The following combinations of information passing are possible:
  1. The caller function passes information and the called function returns information.
  2. The caller function passes information but the called function returns none.
  3. The caller function passes nothing but the called function returns information.
  4. The caller function passes nothing and the called function returns none.
- The name of a function is used in three ways:
  1. for declaration
  2. in a call
  3. for definition
- A function must be first declared and defined.

Function declaration

- The general format for declaring a function that accepts some arguments and returns some value as a result can be given as:

```
Return_data_type function_name(data_type variable1, data_type variable2,…);
```

```
char convert_to_uppercase(char ch);
float avg(int a, int b);
int find_largest(int a, int b, int c);
double multiply(float a, float b);
void swap(int a, int b);
void print(void);
```

- ➢The `Return_data_type` to be returned (namely void, int, float, char, double). If return no value then use void.
- ➢The `function_name` (which requires to call the function). The function name is a valid C identifier.
- ➢The `data_type` is similar to the `Return_data_type` but is use to explain each of the variables (e.g. `variable1`, `variable2`,…). If accept no value then use void.

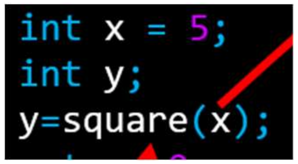
Function definition

- When a function is defined, space is allocated for that function in the memory.
- A function definition comprises two parts:
  1. Function header
  2. Function body
- The syntax of a function definition can be given as:

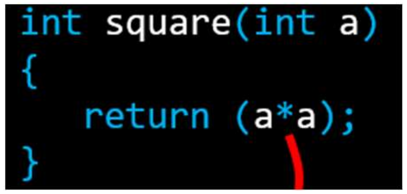
```
Return_data_type function_name(data_type variable1, data_type variable2,…)
{
   ……
   statements
   ……
   return(variable);
}
```

It is the same as the early declaration of a function but without semicolon ;

- Function header is same as that of function declaration. The only difference between the two is that a function header is not followed by a semicolon.
- The list of variables in the function header is known as the formal parameter list. The parameter list may have zero or more parameters of any data type.
- The argument names in the function declaration and function definition need not be the same. However, the data types of the arguments must match with that specified in function declaration as well as function definition.
- The function body is comprising of program statement within {}.



```
int x = 5;
int y;
y=square(x);
```

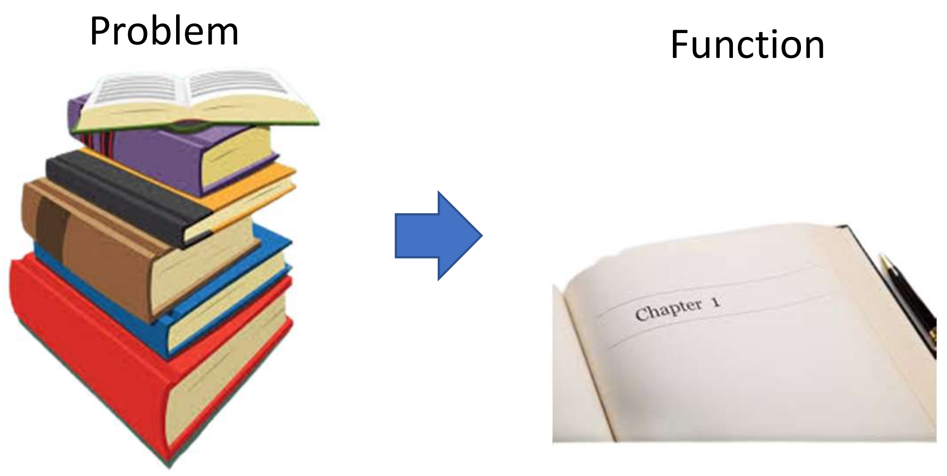


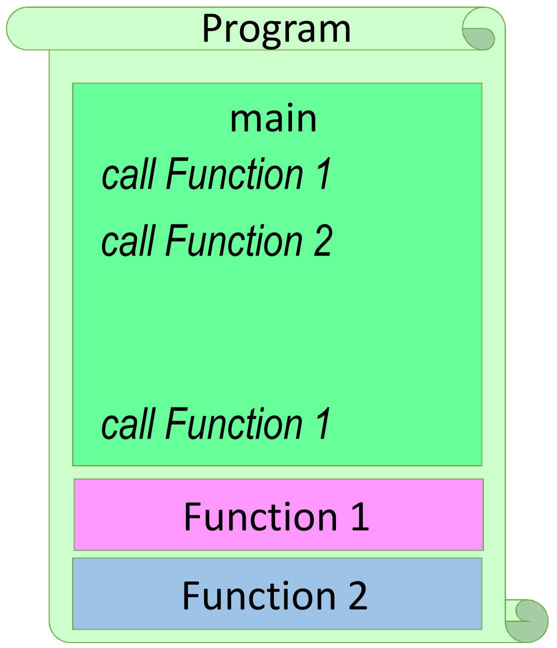
```
int square(int a)
{
    return (a*a);
}
```

ARE YOU OKAY?

Review of today lecture

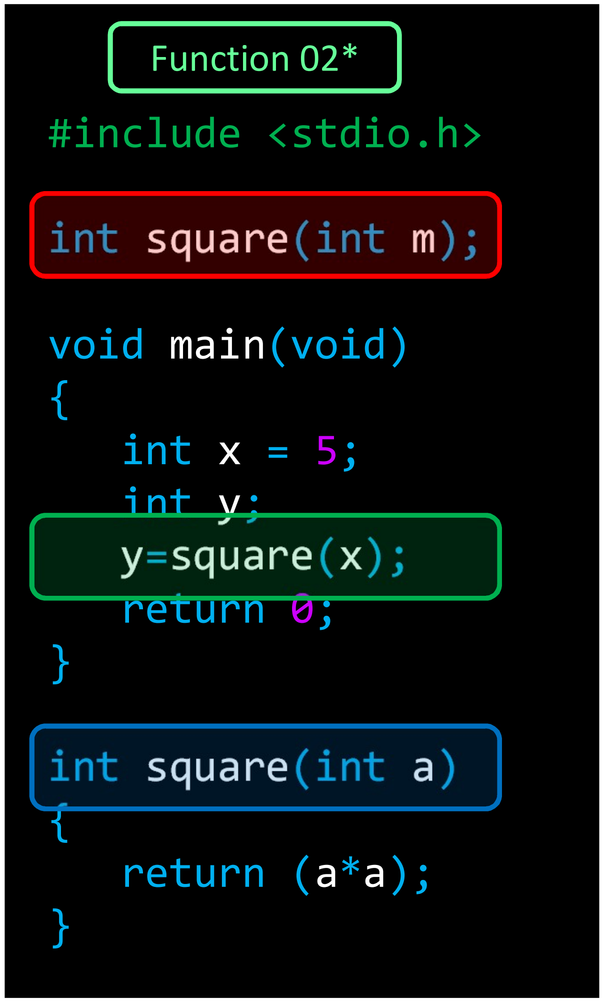
- C program decomposes a program into its component functions.
- Function is a series of statements that have been grouped together and given a name.





Review of today lecture

- The name of a function is used in three ways:
    1. for declaration
    2. in a call
    3. for definition
- A function must be first declared and defined.
- The argument names in the function declaration and function definition need not be the same. However, the data types of the arguments must match with that specified in function declaration as well as function definition.



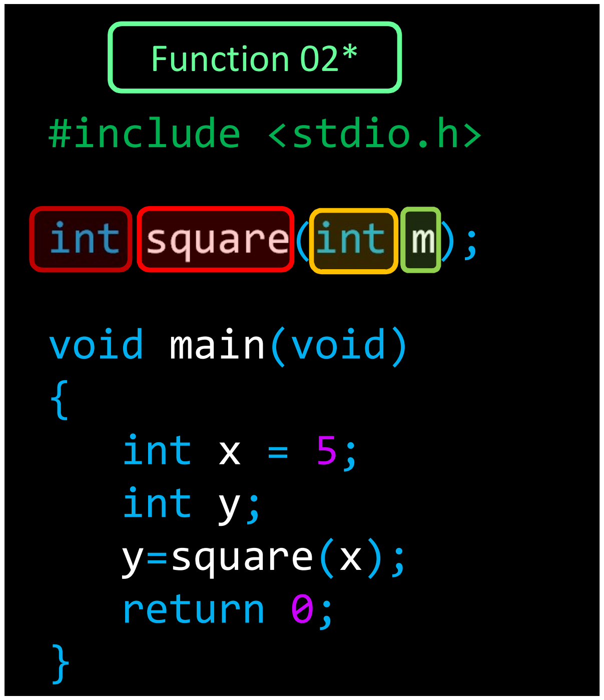
Function 02*

```
#include <stdio.h>

int square(int m);

void main(void)
{
    int x = 5;
    int y;
    y=square(x);
    return 0;
}

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

Review of today lecture

- The syntax of a function declaration can be given as:

```
Return_data_type function_name(data_type variable1, data_type variable2,…);
```



Function 02*

```
#include <stdio.h>

int square(int m);

void main(void)
{
    int x = 5;
    int y;
    y=square(x);
    return 0;
}
```

Don’t forget about semicolon at end of statement for function declaration

Review of today lecture

- The syntax of a function definition can be given as:

```
Return_data_type function_name(data_type variable1, data_type variable2,…)
{
   ……
   statements
   ……
   return(variable);
}
```

Function 02*

```
#include <stdio.h>

int square(int m);

void main(void)
{
    int x = 5;
    int y;
    y=square(x);
    return 0;
}
```

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

Don’t write semicolon at end of statement for function defination