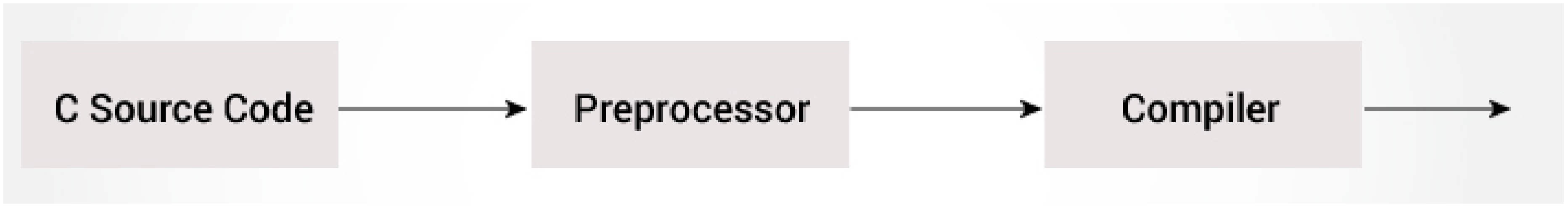
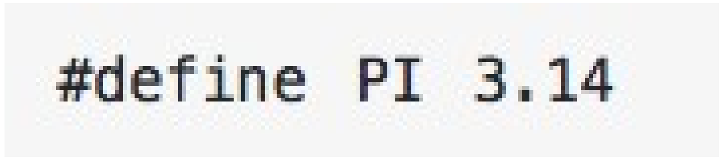
C Programming

Kasun De Zoysa

Preprocessor and Macros



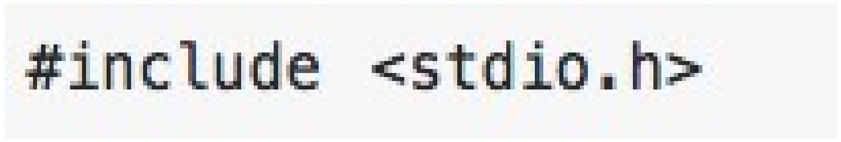
The C preprocessor is a macro preprocessor (allows you to define macros) that transforms your program before it is compiled. All preprocessing directives begins with a # symbol.



```
#define PI 3.14
```

Header Files

The #include preprocessor is used to include header files to a C program.



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

"stdio.h" is a header file. The #include preprocessor directive replaces the above line with the contents of stdio.h header file which contains function and macro definitions.

That's the reason why you need to use #include <stdio.h> before you can use functions like scanf() and printf().

You can also create your own header file containing function declaration and include it in your program.

Macros using #define

A macro is a fragment of code that is given a name. You can use that fragment of code in your program by using the name.

```
#define c 299792458  // speed of light
```

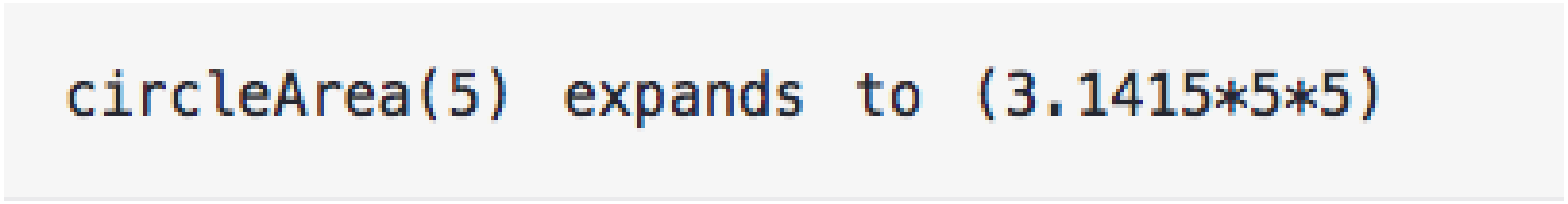
You can also define macros that works like a function call, known as function-like macros.

```
#define circleArea(r) (3.1415*r*r)
```

Macros using #define

Every time the program encounters circleArea(argument), it is replaced by (3.1415*(argument)*(argument)).

Suppose, we passed 5 as an argument then, it expands as below:



```
circleArea(5) expands to (3.1415*5*5)
```

Consider the following Mathematical Function:

A typical exercise in an algebra book asks you to evaluate an expression like

```
n/3+2
```

for n = 2, n = 5, and n = 9.

We can formulate such an expression as a program and use the program as many times as necessary.

Here is the program that corresponds to the above expression:

```
#define f(n) (n/3.0+2)
```

How about the following mathematical functions?

Also formulate the following six expressions as programs:

1. $n^2 + 10$
2. $(n^2 + 40)/2$
3. $2 – (1/n)$
4. $n^2 + (1/2) * n^2 + 30$
5. $(2 – (1/n))*(n^2 + 10)$
6. $(1/2) * n^2 + 20+(2 – (1/n))*(n^2 + 10)$

Determine their results for n = 2 and n = 9.

Conditional Compilation

In C programming, you can instruct preprocessor whether to include certain chuck of code or not. To do so, conditional directives can be used.

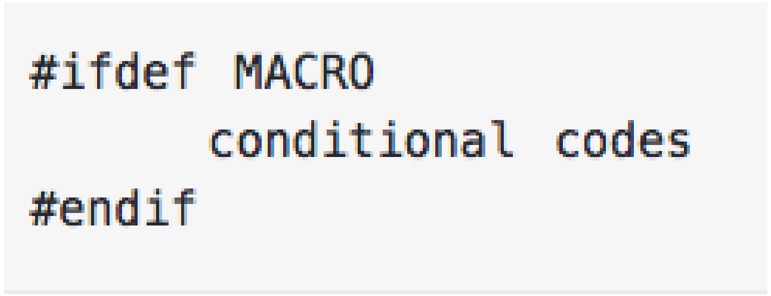
Uses of Conditional

1)use different code depending on the machine, operating system

2)compile same source file in two different programs

3)to exclude certain code from the program but to keep it as reference for future purpose

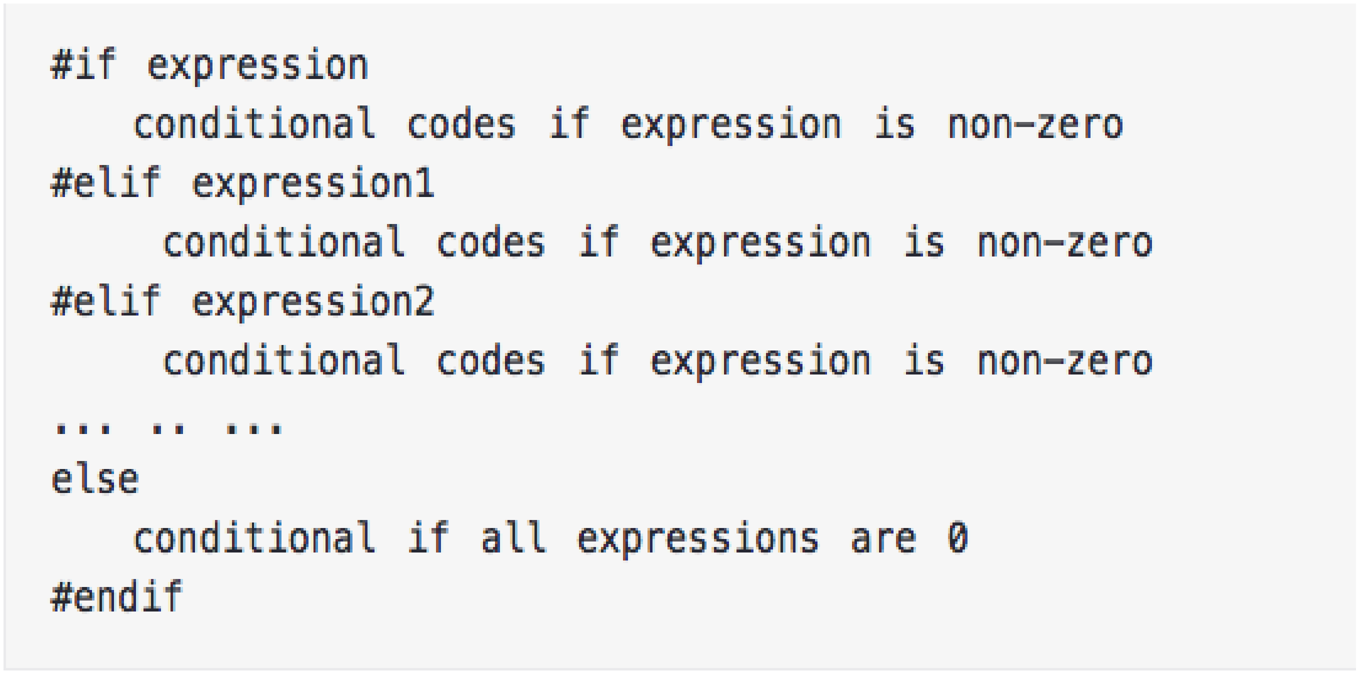
To use conditional, #ifdef, #if, #defined, #else and #elseif directives are used.



```
#ifdef MACRO
      conditional codes
#endif
```

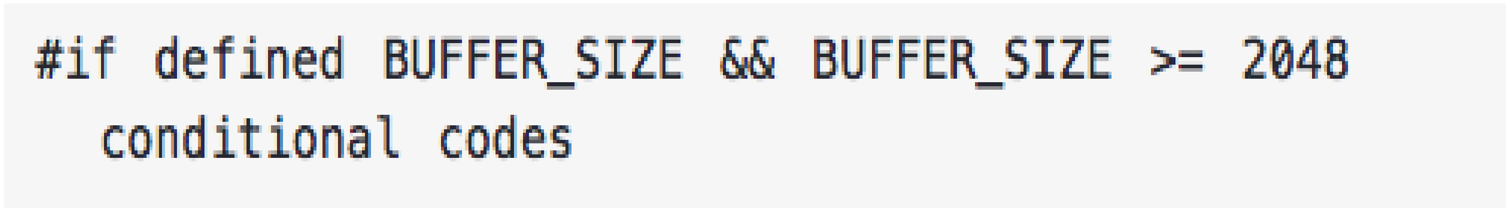
Conditional Compilation

You can also add nested conditional to your #if...#else using #elif



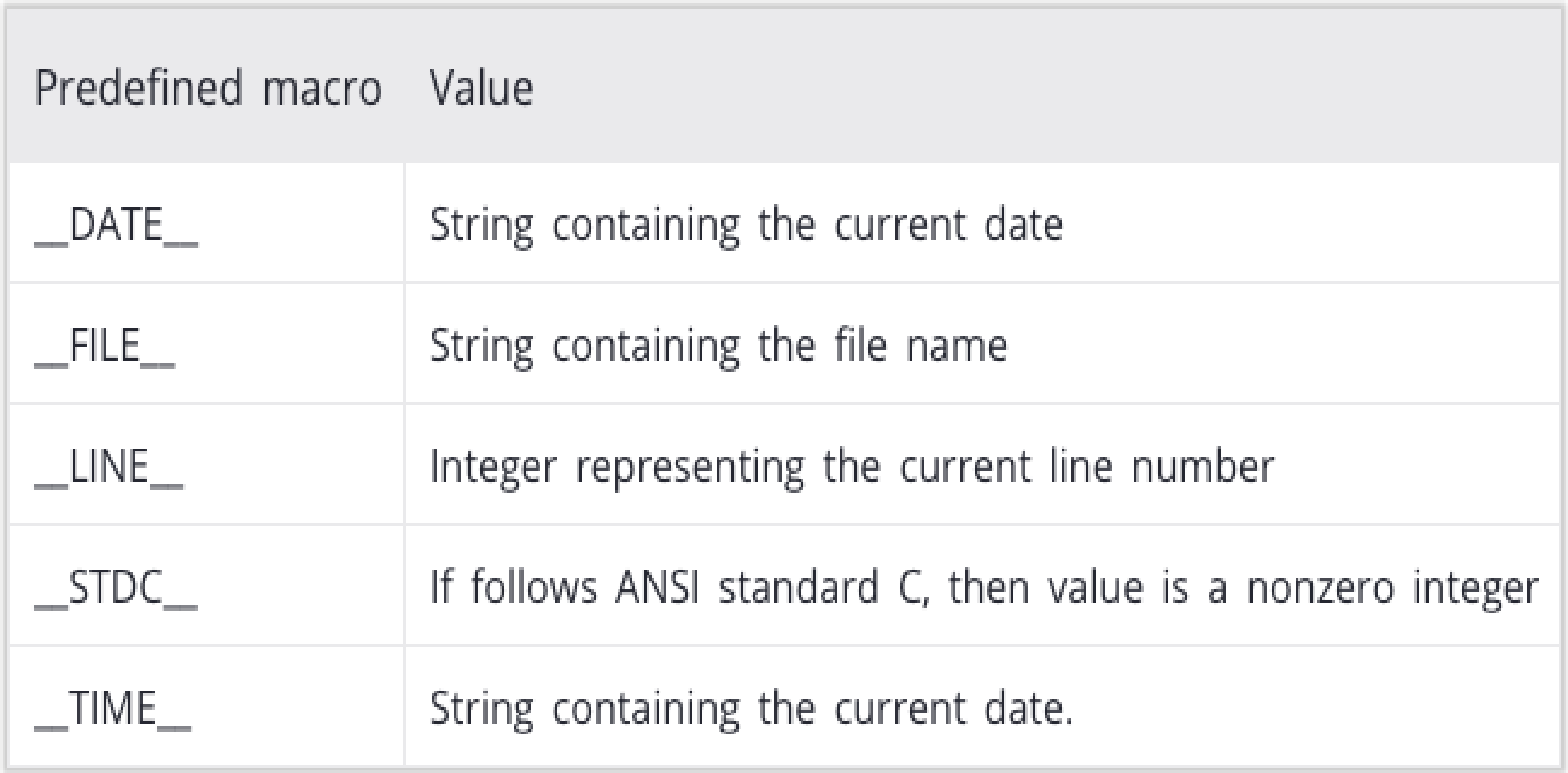
```
#if expression
   conditional codes if expression is non-zero
#elif expression1
    conditional codes if expression is non-zero
#elif expression2
    conditional codes if expression is non-zero
... .. ...
else
   conditional if all expressions are 0
#endif
```

The special operator #defined is used to test whether certain macro is defined or not. It's often used with #if directive.

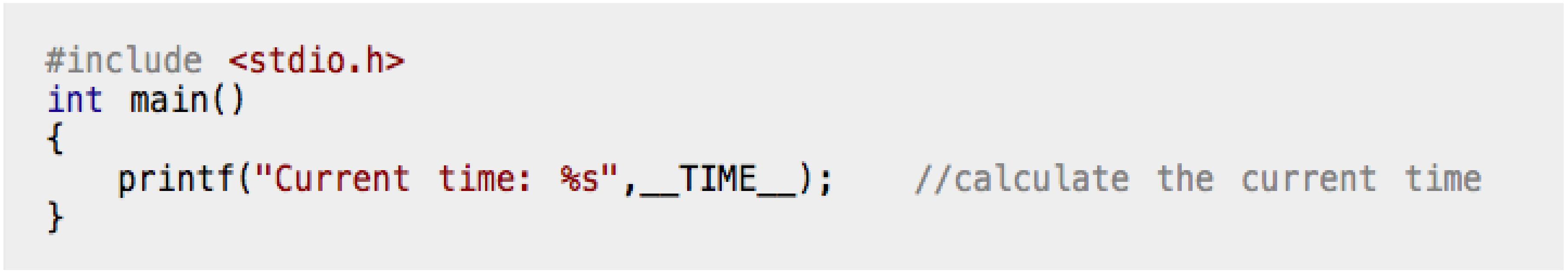


```
#if defined BUFFER_SIZE && BUFFER_SIZE >= 2048
  conditional codes
```

Predefined Macros



Predefined macro	Value
__DATE__	String containing the current date
__FILE__	String containing the file name
__LINE__	Integer representing the current line number
__STDC__	If follows ANSI standard C, then value is a nonzero integer
__TIME__	String containing the current date.



```
#include <stdio.h>
int main()
{
    printf("Current time: %s",__TIME__);    //calculate the current time
}
```

Consider the following problem:

Imagine the owner of a movie theater who has complete freedom in setting ticket prices. The more he charges, the fewer the people who can afford tickets.

In a recent experiment the owner determined a precise relationship between the price of a ticket and average attendance.

At a price of Rs 15.00 per ticket, 120 people attend a performance.
Decreasing the price by 5 Rupees increases attendance by 20 and increasing the price by 5 Rupees decreases attendance by 20.

Unfortunately, the increased attendance also comes at an increased cost. Every performance costs the owner Rs.500. Each attendee costs another 3 Rupees.

The owner would like to know the exact relationship between profit and ticket price so that he can determine the price at which he can make the highest profit.

When we are confronted with such a situation, it is best to tease out the various dependencies one at a time:

1) Profit is the difference between revenue and costs.
2) The revenue is exclusively generated by the sale of tickets. It is the product of ticket price and number of attendees.
3) The costs consist of two parts: a fixed part (Rs.500) and a variable part (Rs. 3 per attendee) that depends on the number of attendees.
4) Finally, the problem statement also specifies how the number of attendees depends on the ticket price.

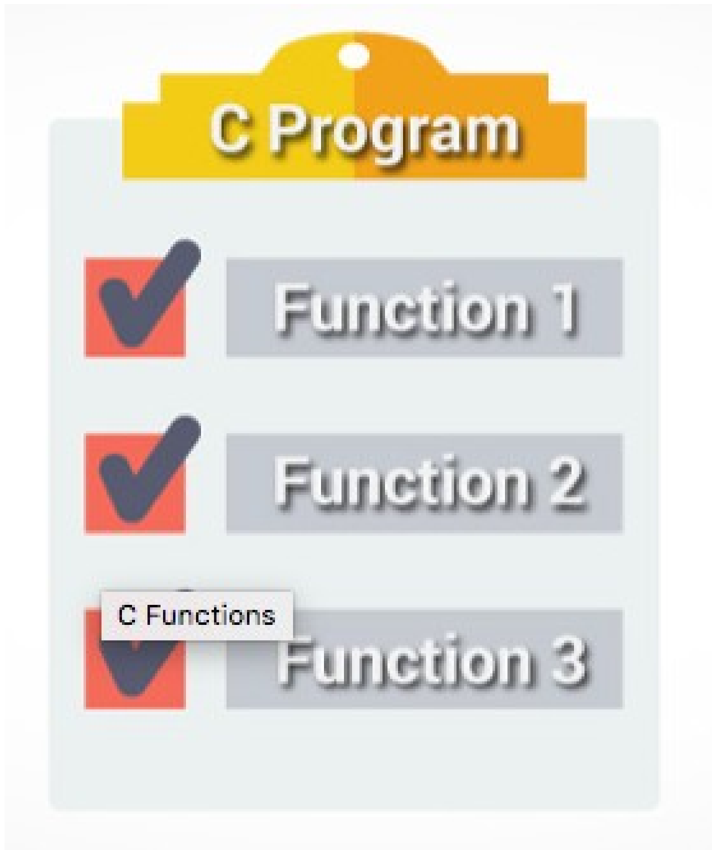
function

A **function** is a block of code that performs a specific task.

Dividing complex problem into small components makes program easy to understand and use.

There are two types of functions in C programming:

- Standard library functions
- User defined functions



The standard library functions

The standard library functions are built-in functions in C programming to handle tasks such as mathematical computations, I/O processing, string handling etc.

These functions are defined in the header file. When you include the header file, these functions are available for use.

For example:
The **printf()** is a standard library function to send formatted output to the screen (display output on the screen). This function is defined in "stdio.h" header file.

There are other numerous library functions defined under "stdio.h", such as scanf(), fprintf(), getchar() etc. Once you include "stdio.h" in your program, all these functions are available for use.

User-defined function

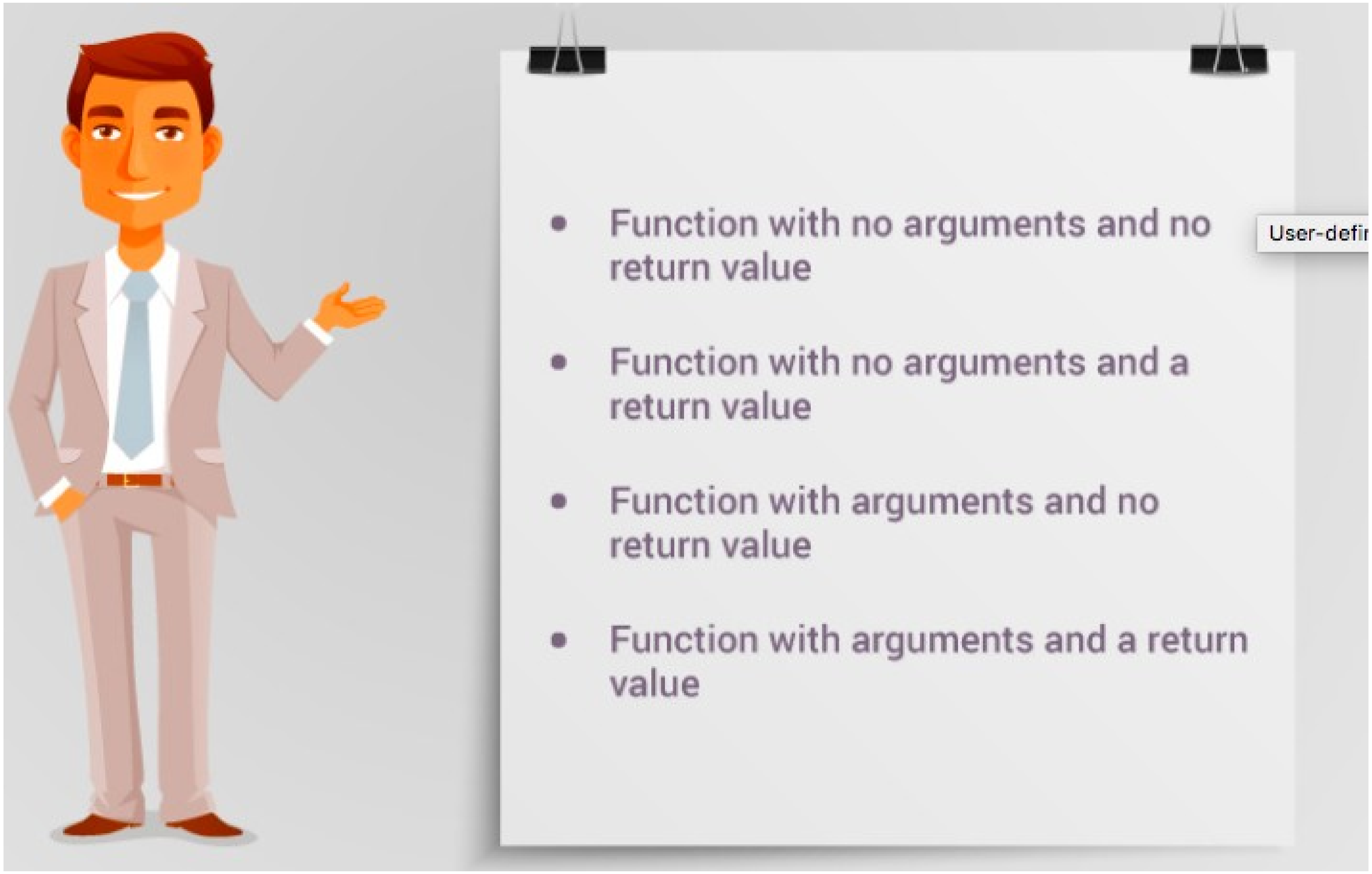
A function is a block of code that performs a specific task.

C allows you to define functions according to your need. These functions are known as user-defined functions.

For example:
Suppose, you need to create a circle and color it depending upon the radius and color. You can create two functions to solve this problem:

```
createCircle() function
color() function
```

Types of User-defined Functions

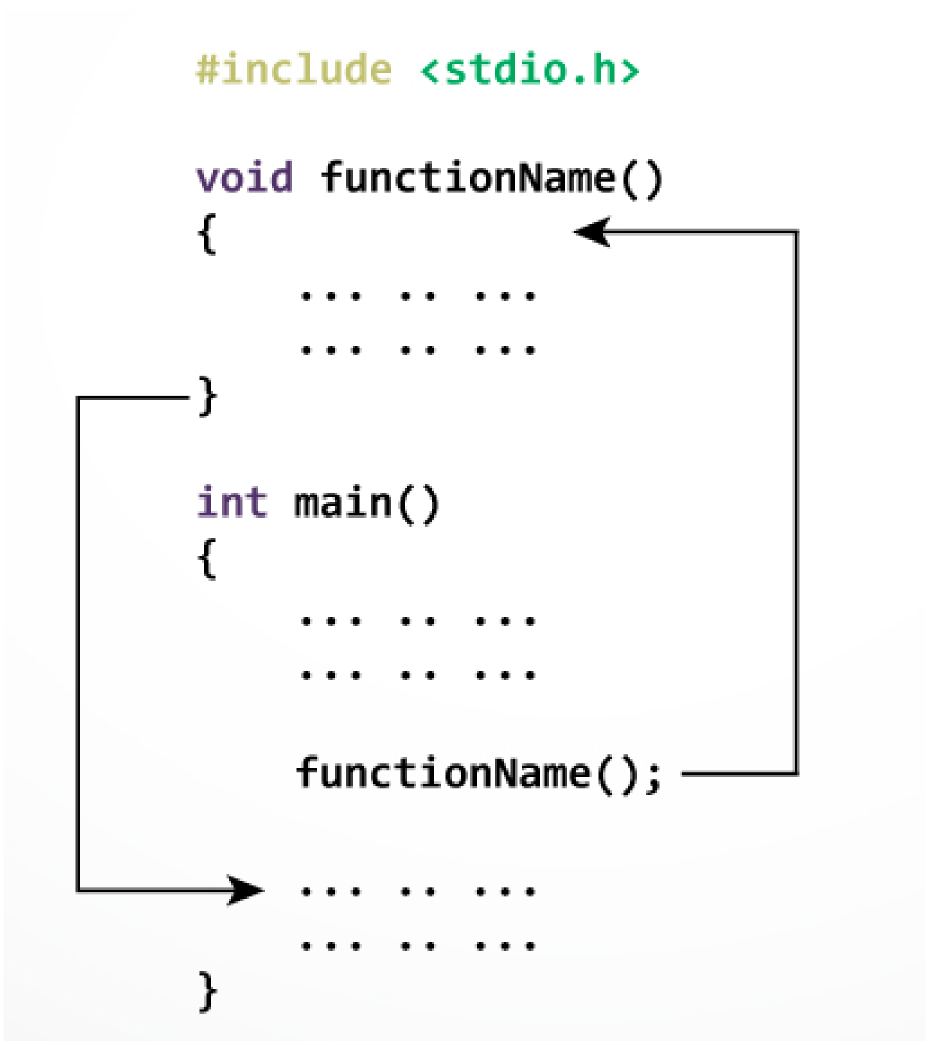


- Function with no arguments and no return value
- Function with no arguments and a return value
- Function with arguments and no return value
- Function with arguments and a return value

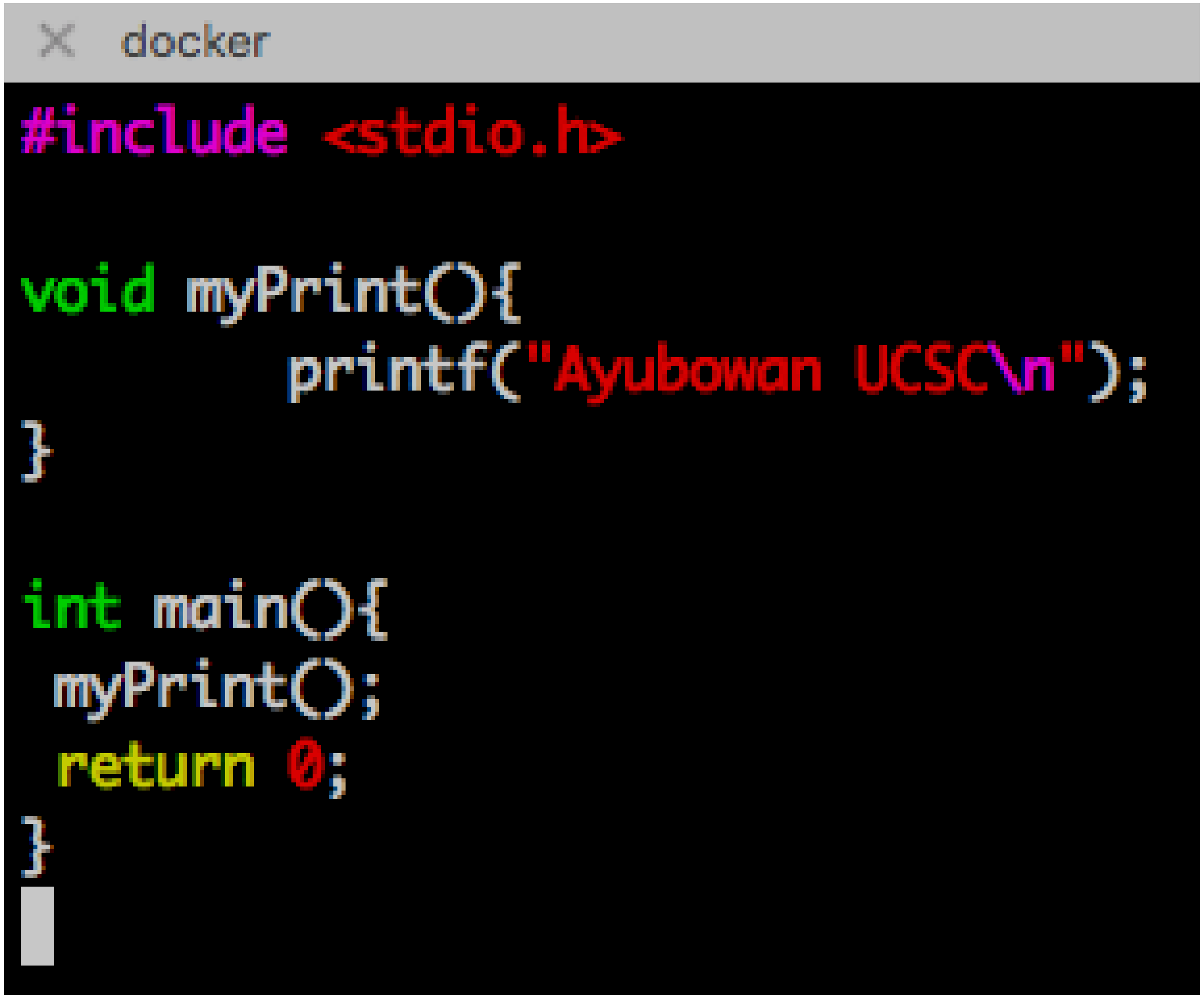
User-defi

How function works?

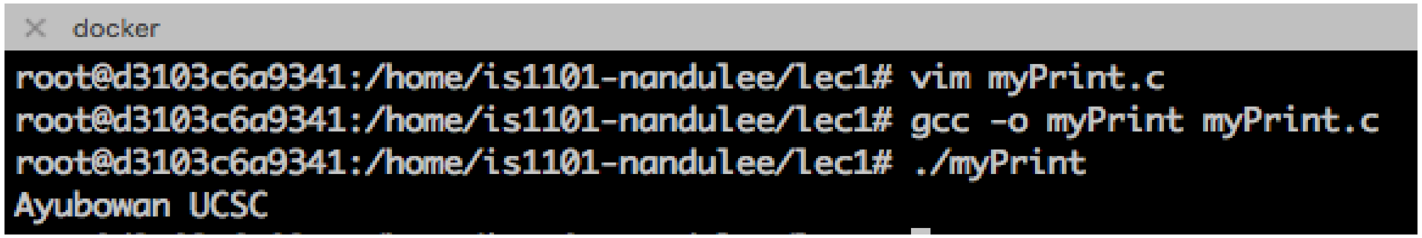
```
#include <stdio.h>
void functionName()
{
    ... .. ...
    ... .. ...
}

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

    functionName();

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



```
#include <stdio.h>

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

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

    functionName();

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



```c
#include <stdio.h>

void myPrint(){
        printf("Ayubowan UCSC\n");
}

int main(){
 myPrint();
 return 0;
}
```



```
root@d3103c6a9341:/home/is1101-nandulee/lec1# vim myPrint.c
root@d3103c6a9341:/home/is1101-nandulee/lec1# gcc -o myPrint myPrint.c
root@d3103c6a9341:/home/is1101-nandulee/lec1# ./myPrint
Ayubowan UCSC
```

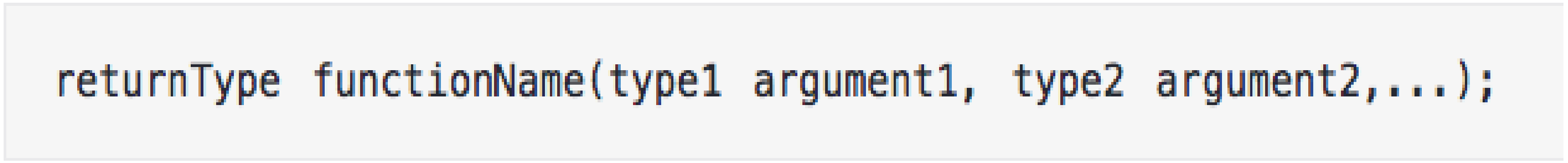
Advantages of function

- The program will be easier to understand, maintain and debug.
- Reusable codes that can be used in other programs
- A large program can be divided into smaller modules.
- Hence, a large project can be divided among many programmers.

A function prototype

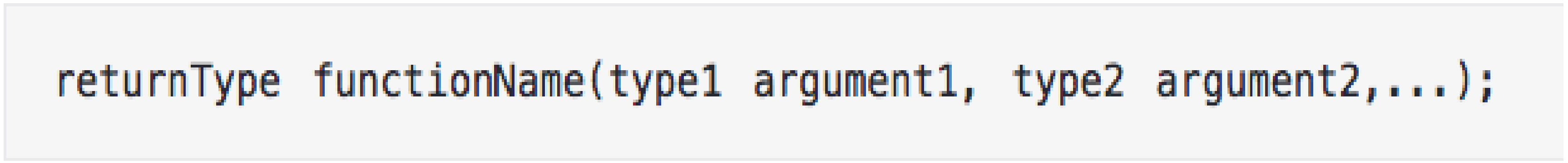
A function prototype is simply the declaration of a function that specifies function's name, parameters and return type. It doesn't contain function body.

A function prototype gives information to the compiler that the function may later be used in the program.



```
returnType functionName(type1 argument1, type2 argument2,...);
```

Add Two Numbers



```
returnType functionName(type1 argument1, type2 argument2,...);
```

for example:

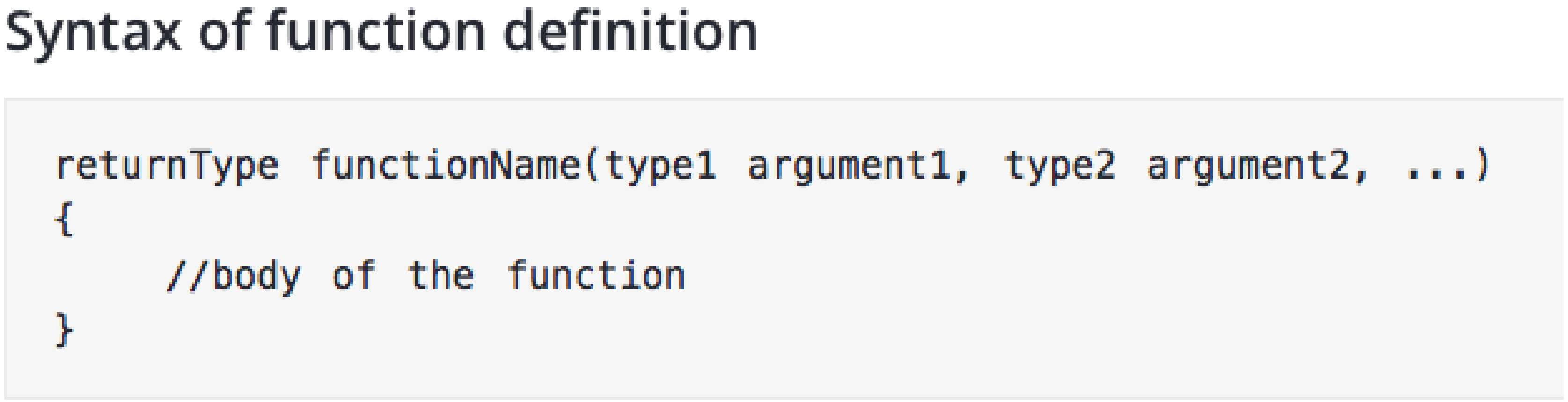
```
int addNumbers(int,int);
```

is the function prototype which provides following information to the compiler:

1. name of the function is addNumbers()
2. return type of the function is int
3. two arguments of type int are passed to the function

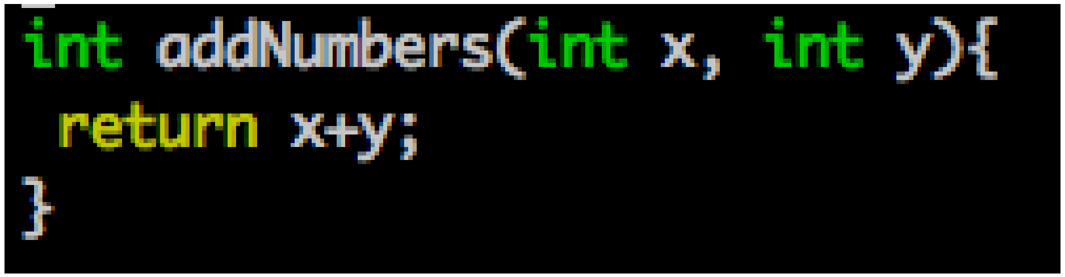
Function Definition

Function definition contains the block of code to perform a specific task i.e. in this case, adding two numbers and returning it.



Syntax of function definition

```
returnType functionName(type1 argument1, type2 argument2, ...)
{
      //body of the function
}
```



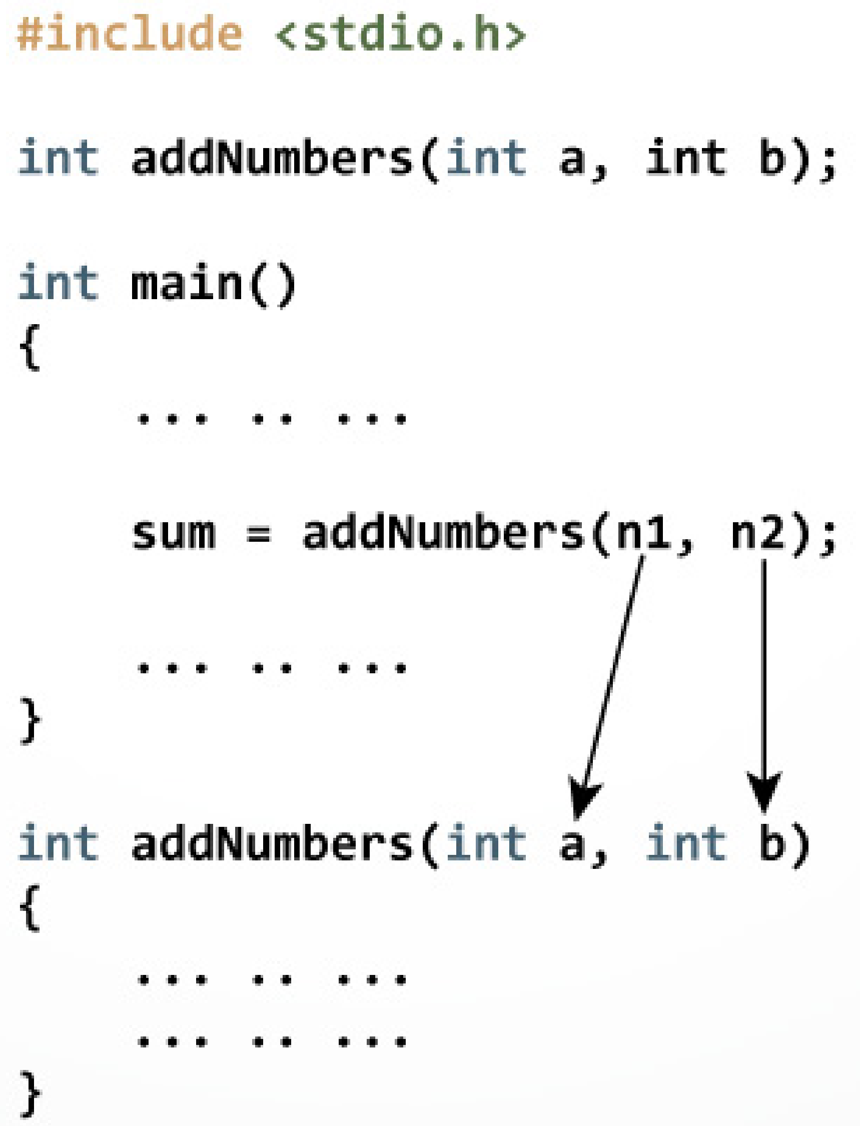
```
int addNumbers(int x, int y){
 return x+y;
}
```

Passing arguments to a function

In programming, argument refers to the variable passed to the function. In the above example, two variables n1 and n2 are passed during function call.

The parameters a and b accepts the passed arguments in the function definition. These arguments are called formal parameters of the function.

How to pass arguments to a function?

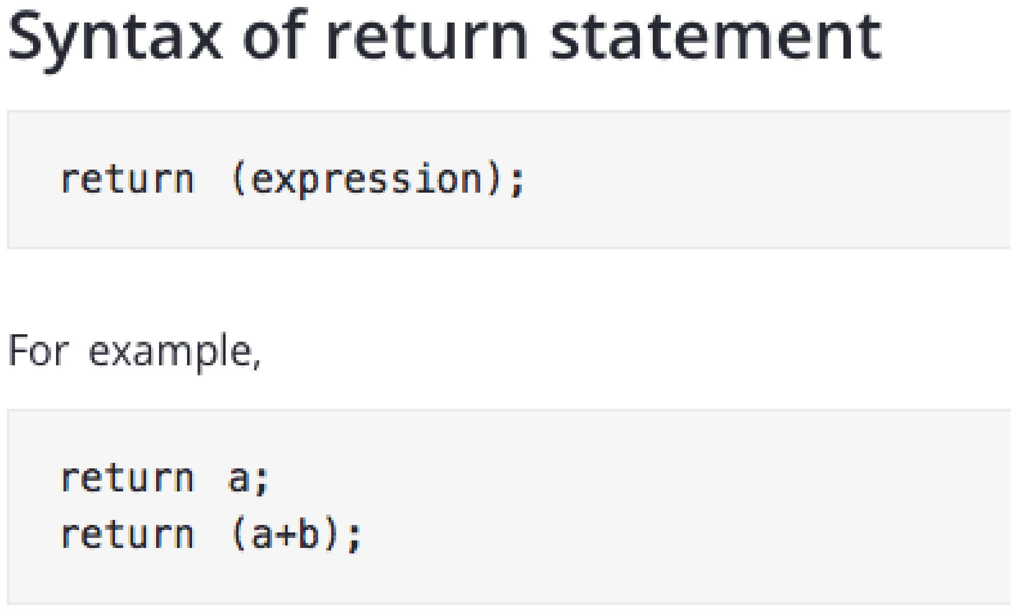


```
#include <stdio.h>

int addNumbers(int a, int b);

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

    sum = addNumbers(n1, n2);

    ... .. ...
}

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

Return statement

The return statement terminates the execution of a function and returns a value to the calling function. The program control is transferred to the calling function after return statement.

In the above example, the value of variable result is returned to the variable sum in the main() function.



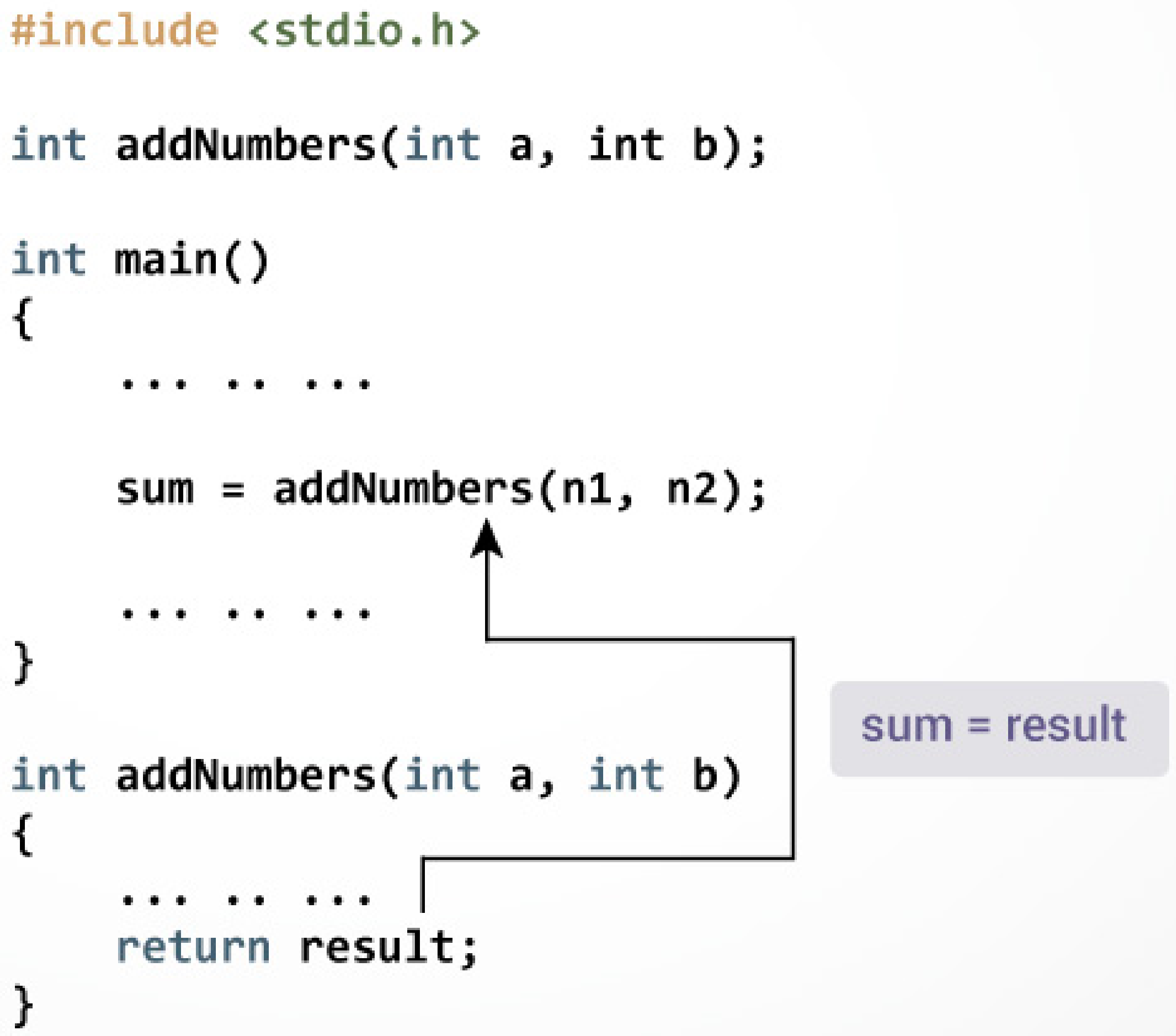
Syntax of return statement

```
return (expression);
```

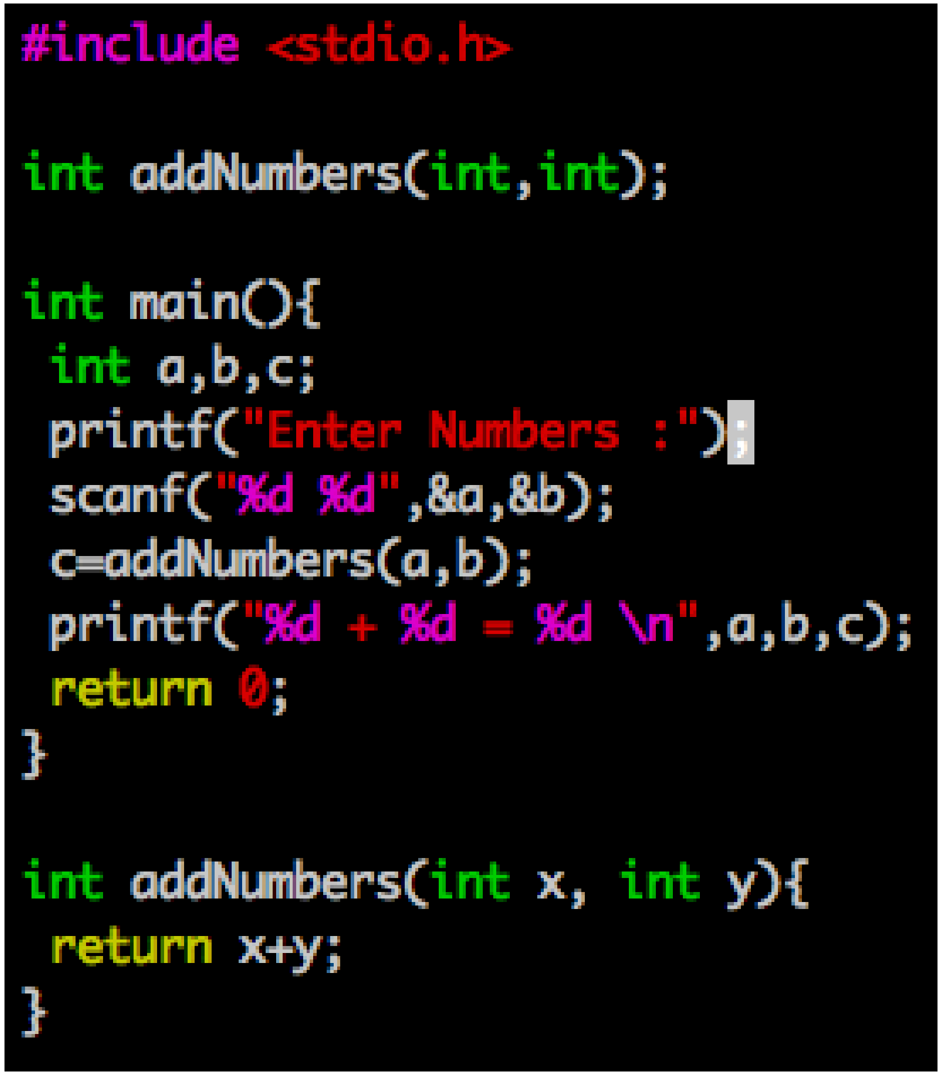
For example,

```
return a;
return (a+b);
```

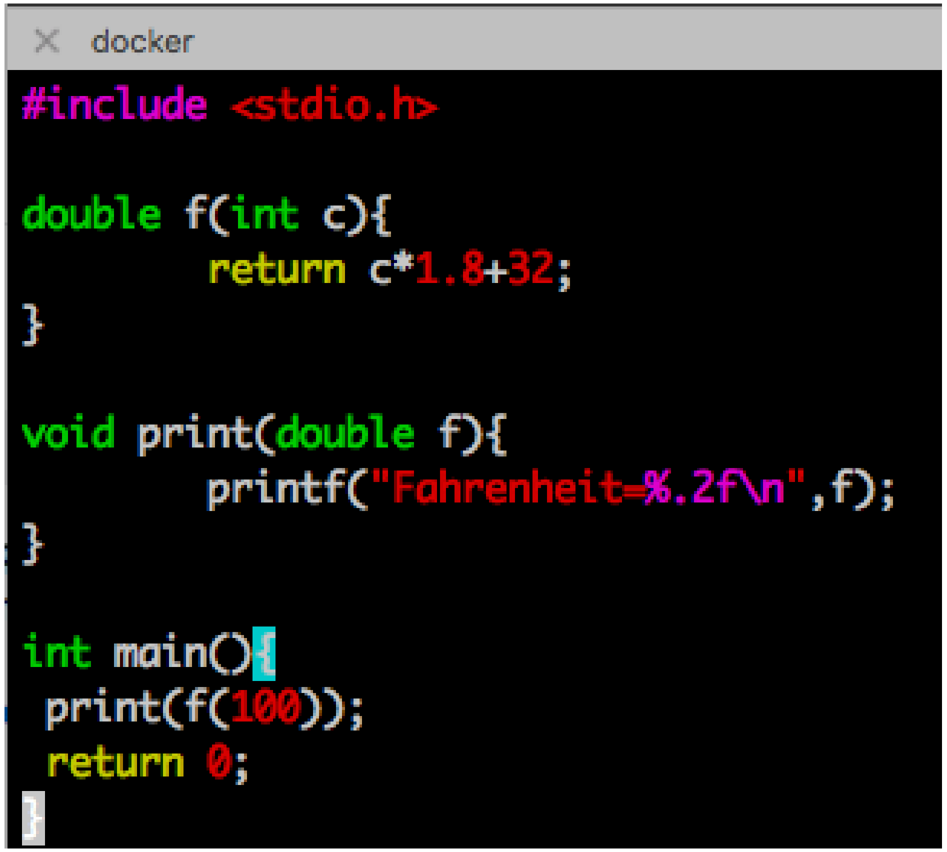
Return statement of a Function



```
#include <stdio.h>

int addNumbers(int a, int b);

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

    sum = addNumbers(n1, n2);

    ... .. ...
}

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



```c
#include <stdio.h>

int addNumbers(int,int);

int main(){
 int a,b,c;
 printf("Enter Numbers :");
 scanf("%d %d",&a,&b);
 c=addNumbers(a,b);
 printf("%d + %d = %d \n",a,b,c);
 return 0;
}

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

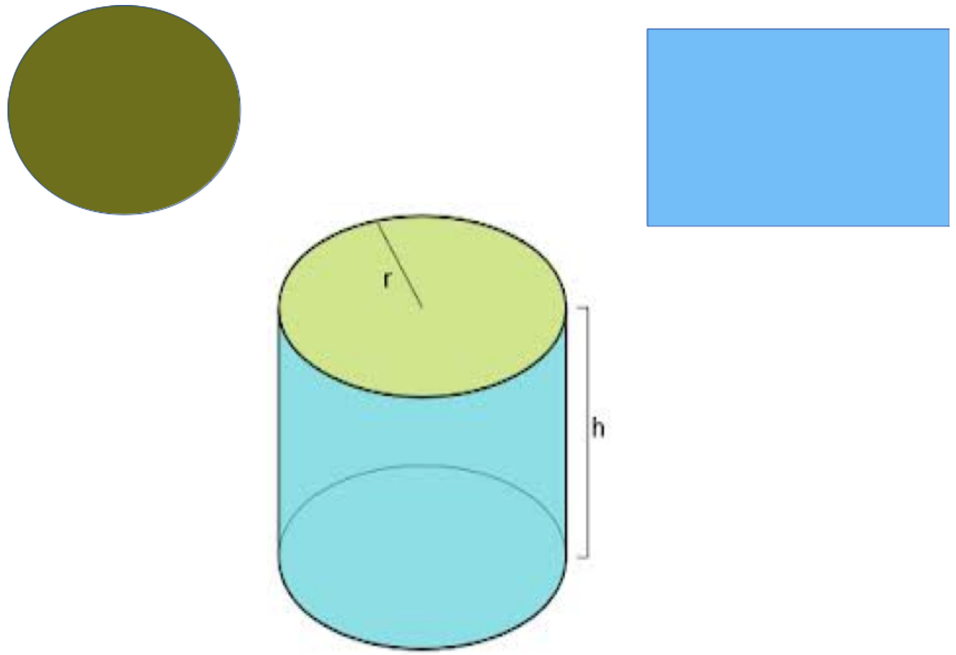


docker

```c
#include <stdio.h>

double f(int c){
        return c*1.8+32;
}

void print(double f){
        printf("Fahrenheit=%.2f\n",f);
}

int main(){
 print(f(100));
 return 0;
}
```

Problem -1

1)Write a program to calculate area of a cylinder by using macros.



Problem -2

Company XYZ & Co. pays all its employees Rs.150 per normal working hour and Rs. 75 per OT hour. A typical employee works 40 (normal) and 20(OT) hours per week has to pay 10% tax. Develop a program that determines the take home salary of an employee from the number of working hours and OT hours given.

Normal Rate Rs.150 per hour
OT rate Rs. 75 per hour.

Total income= wage + OT
TAX = total income * 10%.

So what is the take home salary?