PDS

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

Suppose, you need to write a program to calculate the area of a circle and and area of a rectangle. You can create two functions to solve this problem:

- Write function to calculate area of a circle
- Write function to calculate area of a rectangle

Dividing a complex problem into smaller chunks makes our program easy to understand and reuse.

Types of function

There are two types of function in C programming:

- **Standard library functions**
- **User-defined functions**

Standard library functions

The standard library functions are built-in functions in C programming.

These functions are defined in header files. 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 the stdio.h header file. Hence, to use the printf()function, we need to include the stdio.h header file using #include <stdio.h>.
- The sqrt() function calculates the square root of a number. The function is defined in the math.h header file.

User-defined function

You can also create functions as per your need. Such functions created by the user are known as user-defined functions.

How user-defined function works?

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

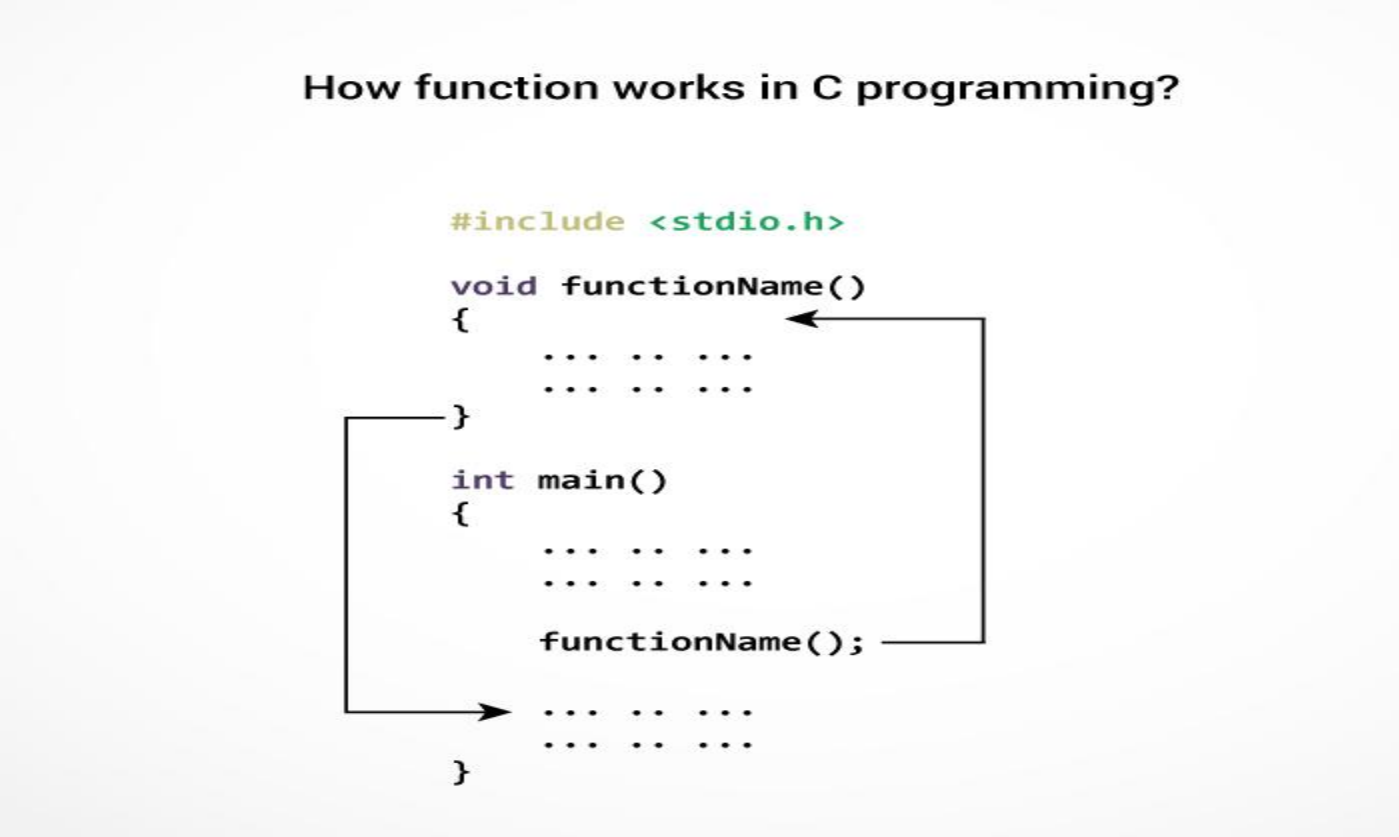
The execution of a C program begins from the main() function.

When the compiler encounters functionName();, control of the program jumps to

void functionName()

And, the compiler starts executing the codes inside functionName()

The control of the program jumps back to the main() function once code inside the function definition is executed.



Note, function names are identifiers and should be unique.

For better understanding of arguments and return value from the function, user-defined functions can be categorized as:

- 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.

The 4 programs below check whether an integer entered by the user is a prime number or not. And, all these programs generate the same output.

Example #1: No arguments passed and no return Value

```
#include <stdio.h>

void checkPrimeNumber();

int main()

{

   checkPrimeNumber();    // no argument is passed to prime()
```

```
    return 0;
}
// return type of the function is void becuase no value is returned from the function
void checkPrimeNumber()
{
    int n, i, flag=0;
    printf("Enter a positive integer: ");
    scanf("%d",&n);
    for(i=2; i <= n/2; ++i)
    {
        if(n%i == 0)
        {
            flag = 1;
        }
    }
    if (flag == 1)
        printf("%d is not a prime number.", n);
    else
        printf("%d is a prime number.", n);
}
```

The checkPrimeNumber() function takes input from the user, checks whether it is a prime number or not and displays it on the screen.

The empty parentheses in checkPrimeNumber(); statement inside the main() function indicates that no argument is passed to the function.

The return type of the function is void. Hence, no value is returned from the function.

Example #2: No arguments passed but a return value

```
#include <stdio.h>
int getInteger();
int main()
```

```
{
    int n, i, flag = 0;
    // no argument is passed to the function
    // the value returned from the function is assigned to n
    n = getInteger();
    for(i=2; i<=n/2; ++i)
    {
        if(n%i==0){
            flag = 1;
            break;
        }
    }
    if (flag == 1)
        printf("%d is not a prime number.", n);
    else
        printf("%d is a prime number.", n);
    return 0;
}
// getInteger() function returns integer entered by the user
int getInteger()
{
    int n;
    printf("Enter a positive integer: ");
    scanf("%d",&n);
    return n;
}
```

The empty parentheses in n = getInteger(); statement indicates that no argument is passed to the function. And, the value returned from the function is assigned to n.

Here, the getInteger() function takes input from the user and returns it. The code to check whether a number is prime or not is inside the main() function.

Example #3: Argument passed but no return value

```
#include <stdio.h>
void checkPrimeAndDisplay(int n);
int main()
{
    int n;
    printf("Enter a positive integer: ");
    scanf("%d",&n);
    // n is passed to the function
    checkPrimeAndDisplay(n);
    return 0;
}
// void indicates that no value is returned from the function
void checkPrimeAndDisplay(int n)
{
    int i, flag = 0;
    for(i=2; i <= n/2; ++i)
    {
        if(n%i == 0){
            flag = 1;
            break;
        }
    }
    if(flag == 1)
        printf("%d is not a prime number.",n);
    else
        printf("%d is a prime number.", n);
}
```

The integer value entered by the user is passed to checkPrimeAndDisplay() function.

Here, the checkPrimeAndDisplay() function checks whether the argument passed is a prime number or not and displays the appropriate message.

Example #4: Argument passed and a return value

```
#include <stdio.h>
int checkPrimeNumber(int n);
int main()
{
   int n, flag;
   printf("Enter a positive integer: ");
   scanf("%d",&n);
   // n is passed to the checkPrimeNumber() function
   // the value returned from the function is assigned to flag variable
   flag = checkPrimeNumber(n);
   if(flag==1)
      printf("%d is not a prime number",n);
   else
      printf("%d is a prime number",n);
   return 0;
}
// integer is returned from the function
int checkPrimeNumber(int n)
{
   /* Integer value is returned from function checkPrimeNumber() */
   int i;
   for(i=2; i <= n/2; ++i)
   {
      if(n%i == 0)
         return 1;
   }
```

```
    return 0;
}
```

The input from the user is passed to checkPrimeNumber() function.

The checkPrimeNumber() function checks whether the passed argument is prime or not. If the passed argument is a prime number, the function returns 0. If the passed argument is a non-prime number, the function returns 1. The return value is assigned to flag variable.

Then, the appropriate message is displayed from the main() function.

Which approach is better?

Well, it depends on the problem you are trying to solve. In case of this problem, the last approach is better.

A function should perform a specific task. The checkPrimeNumber() function doesn't take input from the user nor it displays the appropriate message. It only checks whether a number is prime or not, which makes code modular, easy to understand and debug.