```
CSE1062 | CCS1063 'Practicals' {

    [Fundamentals of Computer Programming]

        < Tutorial Session 06 - Functions >

}
```

What is Function?

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

Suppose, you need to create a program to create a circle and color it. You can create two functions to solve this problem:

* create a circle function
* create a color function

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

https://www.programiz.com/c-programming/c-functions

Types of Functions in C

There are two types of function in C programming:

1. **Standard library functions**
2. **User-defined functions**

https://www.programiz.com/c-programming/c-functions

Standard library functions

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

These functions are defined in header files.

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.

Visit standard library functions in C programming to learn more.

https://www.programiz.com/c-programming/c-functions

User-defined function

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

Advantages of user-defined 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.

https://www.programiz.com/c-programming/c-functions

How user-defined function works?

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

https://www.programiz.com/c-programming/c-functions

Rules

* Function may accept as many parameters as it needs or no parameters.
* Function may return either one or no values.
* Variables declared inside a function are only available to that function.

Function Prototype

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

```
int add_positive_numbers(int a, int b);
int addPstvNmbrs(int a, int b);
```

https://www.programiz.com/c-programming/c-user-defined-functions

Function Definition

Function definition contains the block of code to perform a specific task. In our example, adding two numbers and returning it.

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

//body of the function
}
```

https://www.programiz.com/c-programming/c-user-defined-functions

Calling Function

Control of the program is transferred to the user-defined function by calling it. In the above example, the **function call** is made using **addNumbers(n1, n2);** statement inside the **main() function**.

```
functionName(argument1,argument2,...);
```

https://www.programiz.com/c-programming/c-user-defined-functions

Passing arguments

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

```
addNumbers( n1, n2);
```

https://www.programiz.com/c-programming/c-user-defined-functions

```c
#include <stdio.h>
int main(){
    …… …. …… …
    sum = addNumbers( n1, n2);
   …… …. …. …

}

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

FUNCTION CALLING

FUNCTION DEFINITION

https://www.programiz.com/c-programming/c-user-defined-functions

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 the return statement. >

Ex:

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

https://www.programiz.com/c-programming/c-user-defined-functions

```
1
2   #include <stdio.h>
3   int main(){
4       …… ….  …… …
5       sum = addNumbers( n1, n2);
6      …….  …. …. …
7                   sum = result
8   }
9
10  int addNumbers(int a , int b)
11  {
12
13          ……… ..  …..  ……
14          return result;
    }
```

FUNCTION CALLING

FUNCTION DEFINITION

https://www.programiz.com/c-programming/c-user-defined-functions

```c
#include <stdio.h>
int addNumbers(int a, int b);
int main(){
   int n1,n2,sum;
    printf("Enters two numbers: ");
    scanf("%d %d",&n1,&n2);
    sum = addNumbers(n1, n2);
    printf("sum = %d", sum);
    return 0;
}
int addNumbers(int a, int b) {
    int result;
    result = a+b;
    return result;
}
```

FUNCTION PROTOTYPE

FUNCTION CALLING

FUNCTION DEFINITION

https://www.programiz.com/c-programming/c-user-defined-functions

Types of User Defined Functions

In Programming we have 04 types of user defined functions.

- No Argument Passed and No Return Value
- No Arguments Passed But Returns a Value
- Argument Passed But No Return Value
- Argument Passed and Returns a Value

Let’s take an example of a program used to **check whether the integer entered by the user is a prime number or not.**

https://www.programiz.com/c-programming/types-user-defined-functions

No Argument Passed and No Return Value

Return type is void meaning doesn't return any value

No Argument is passed

```c
#include <stdio.h>
void checkPrimeNumber();
int main() {
  checkPrimeNumber();
  return 0;
}
void checkPrimeNumber() {

  int n, i, flag = 0;
  printf("Enter a positive integer: ");
  scanf("%d",&n);

// 0 and 1 are not prime numbers
  if (n == 0 || n == 1)
    flag = 1;

  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);
}
```

https://www.programiz.com/c-programming/types-user-defined-functions

No Arguments Passed But Returns a Value

No Argument is passed

Returns integer entered by the user

```c
#include <stdio.h>

int getInteger();

int main() {
  int n, i, flag = 0;
   n = getInteger();

if (n == 0 || n == 1)
    flag = 1;

  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;
}
int getInteger() {

  int n;
  printf("Enter a positive integer: ");
  scanf("%d",&n);

  return n;
}
```

https://www.programiz.com/c-programming/types-user-defined-functions

Argument Passed But No Return Value

Argument is passed

Return type is void meaning doesn't return any value

```c
#include <stdio.h>
void checkPrimeAndDisplay(int n);

int main() {
  int n;
  printf("Enter a positive integer: ");
  scanf("%d",&n);

  checkPrimeAndDisplay(n);

  return 0;
}

void checkPrimeAndDisplay(int n) {

  int i, flag = 0;

  if (n == 0 || n == 1)
    flag = 1;

  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);
}
```

https://www.programiz.com/c-programming/types-user-defined-functions

Arguments Passed and Returns a Value

The returned value is assigned to the flag variable

N is passed to the checkprimenumber() function

```c
#include <stdio.h>
int checkPrimeNumber(int n);

int main() {
  int n, flag;
  printf("Enter a positive integer: ");
  scanf("%d",&n);

  flag = checkPrimeNumber(n);

  if(flag == 1)
    printf("%d is not a prime number",n);
  else
    printf("%d is a prime number",n);

  return 0;
}

int checkPrimeNumber(int n) {

  if (n == 0 || n == 1)
    return 1;

  int i;

  for(i=2; i <= n/2; ++i) {
    if(n % i == 0)
      return 1;
  }

  return 0;
}
```

https://www.programiz.com/c-programming/types-user-defined-functions

Recursive Functions

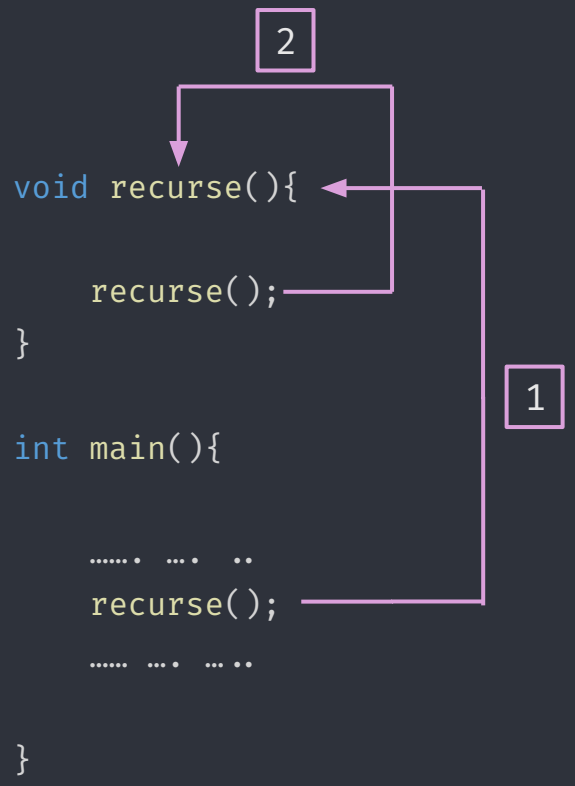
< The C programming language allows any of its functions to call **itself** multiple times in a program. Here, any function that happens to call itself **again and again** (directly or indirectly), unless the program satisfies some **specific condition/subtask** is called a recursive function. >

https://www.programiz.com/c-programming/c-recursion

How Recursive Functions Works?

The recursion continues until some condition is met to prevent it.

https://www.programiz.com/c-programming/c-recursion



Example: Sum of Natural Numbers Using Recursion.

Initially, the sum() is called from the main() function with number passed as an argument.

https://www.programiz.com/c-programming/c-recursion

```c
#include <stdio.h>
int sum(int n);

int main() {
    int number, result;

    printf("Enter a positive integer: ");
    scanf("%d", &number);

    result = sum(number);

    printf("sum = %d", result);
    return 0;
}

int sum(int n) {
    if (n ≠ 0)
        // sum() function calls itself
        return n + sum(n-1);
    else
        return n;
}
```

```c
int main() {
…..
    result = sum(number);
…..
}

int sum(int n) {
    if (n ≠ 0)
        return n + sum(n-1);
    else
        return n;
}
int sum(int n) {
    if (n ≠ 0)
        return n + sum(n-1);
    else
        return n;
}
```

3

3 + 3 = 6 is returned

2

2 + 1 = 3 is returned

1

```c
int sum(int n) {
    if (n ≠ 0)
        return n + sum(n-1);
    else
        return n;
}

int sum(int n) {
    if (n ≠ 0)
        return n + sum(n-1);
    else
        return n;
}
```

0

1 + 0 = 1 is returned

0 is returned

https://www.programiz.com/c-programming/c-recursion

```
Thanks; {

    'Do you have any questions?'

        < bgamage@sjp.ac.lk >

}
```