COMPUTER ENGINEERING

Laboratory
Multimedia and Internet of Things
Computer Engineering Department
Institut Teknologi Sepuluh Nopember

Basic Programming Practicum

Functions and Recursion

2024

1 Goals

- Students are able to create and call functions in C .
- Students are able to pass parameter by value and by reference in C.
- Students understand and are able to apply recursion in C.

2 Function

A function is a collection of statment that is used to perform a spesific task, it may or may not use an input to generate the desired output. These are the advantages of using functions in C programming language are:

- Some code snippets are reusable when using functions.
- C functions can be called any number of times in a program and at any place in a program.
- A complex and large C codes can be splitted to several function, thus easier to track.

2.1 Function Declaration

Every C program has atleast one function, which is the main() function. You can also define functions other than main(). Syntax :

```
return_type function_name( parameters list){
    // function body
 return something;
}
```

- Return Type.
  The data type a function has to return.
- `function_name`.
  The name of the function
- parameters list.
  The parameters of the function.
- Function body.
  The block of code (Statements) that will be executed when the function is called.
- `return something;`
  A statement to return a value (`something`) from the function `function_name`. Returning causes the program to break out of the function. For functions that doesn't return a value (`void` type function), to break out of a function simply write `return;`

Example

```c
float TriangleArea(float Base, float Height)
{
  float Area;
  Area = 0.5*Base*Height;
  return Area;
}
```

2.2 Calling a Function

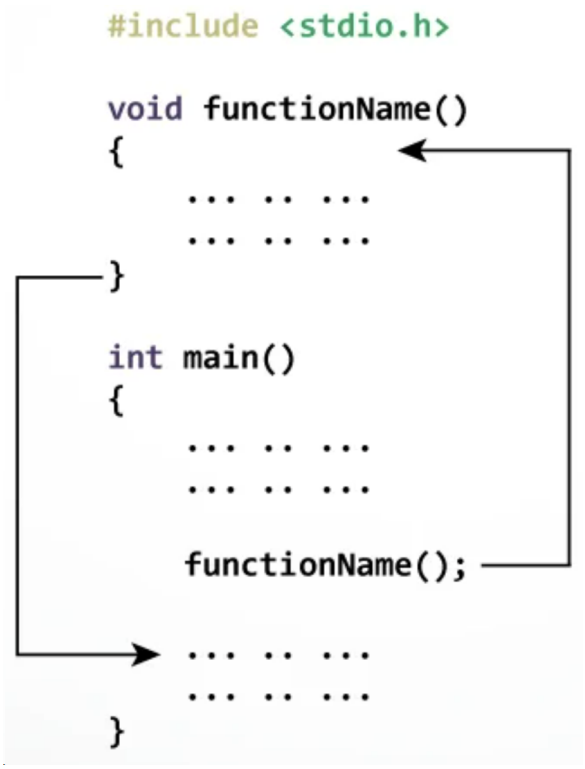

Figure 1

```c
  #include <stdio.h>
  // Declaring a function to calculate a Triangle Area called TriangleArea
% // Mendeklarasikan fungsi luasSegitiga
  // The parameters are the  value of Base and Height
% // Parameter input Alas , dan Tinggi
  // The output data type is float+
% // Output float
float TriangleArea(float Base, float Height)
{
  float Area;
  Area = 0.5*Base*Height;
  return Area;
}
int main()
{
  float Bs = 4,Hg=10,L;
    % // Memanggil Fungsi TriangleArea
  // Calling the TriangleArea function
```

```
19      L=TriangleArea(Bs,Hg);
20    printf("Area = %f",L);
21    return 0;
22  }
```

1. Line 5-10: Defining the function, namely `Triangle Area` with
    - Two input parameters :
      `Base` and `Height` with `float` data type.
    - Singular output with `float` data type.

2.3 Function with Arguments

2.4 Arguments

If a function is expected to use arguments, then the variables that acts as the parameters that receive values from these arguments must be declared beforehand.

1. **Parameters :**
    (a) Parameters are the variables in a function that points to a part of the data that is inserted into the function.
    (b) These data are called arguments.
2. **Formal Parameters:**
    (a) Parameters that are written within the function definition is called "Formal Parameters".
    (b) Formal Parameters are always a variable, Actual Parameter however doesn't necessarily has to be a variable.
3. **Actual Parameters:**
    (a) Parameters that are used when calling a function
    (b) Can be a form of numbers, expressions, or another function call.

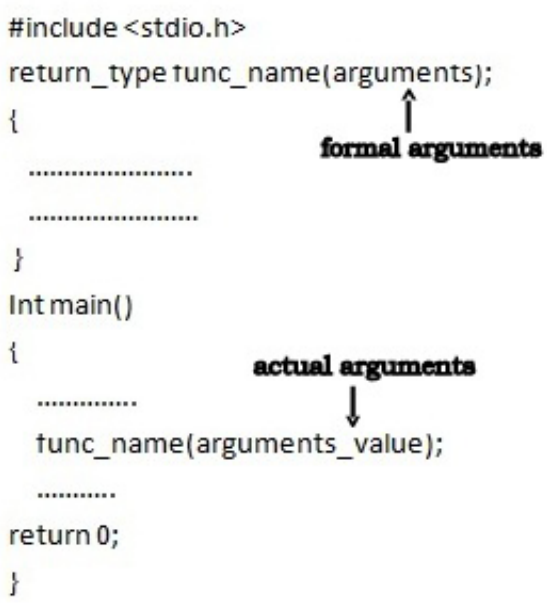

Figure 2

2.5 Parameter Passing

Parameter passing is passing a value to the parameter when calling a function. Generally, there are two ways to pass paramaters into a function :

- Pass parameter by value means to pass the **value** of the variable to the parameter of a function.
- Pass parameter by reference means to pass the reference of a variable **(its memory address)** to the paramater a function.

2.5.1 Passing Parameter by Value

Listing 1: Passing by Value

```
1   #include <stdio.h>
2
3   int swapAndReturnSum(int x, int y) {
4       int z;
5       z = x;
6       x = y;
7       y = z;
8       return x + y;
9   }
10
11  int main() {
12      int a = 1;
13      int b = 2;
14      int sum = swapAndReturnSum(a, b);
15      printf("Sum: %d\n", sum);
16      printf("Value a and b now:\n");
17      printf("a: %d\n", a);
18      printf("b: %d\n", b);
```

```
19          return 0;
20      }
```

Pay attention at the snippet above, line 3-6 of the code in Listing 1 is a set of assignments to swap the values of 2 variables. However, when the program is executed, the output would be the following.

```
sum: 3
values a and b now:
a: 1
b: 2
```

As you can see, the values of a and b did not swap. When passing parameter by value, anything that is done within the function body will have no effect on the parameter that is "passed on" the function. The value of the **actual parameter** will be assigned to the **formal parameter**, so we are not doing operation directly on the actual parameter.

2.5.2 Passing Parameter by Reference

Look at second line of the following code.

Listing 2: Passing by Reference

```
1   #include <stdio.h>
2
3   void swap(int *x, int *y) {
4       int z;
5       z = *x;
6       *x = *y;
7       *y = z;
8   }
9
10  int main() {
11      int a = 1;
12      int b = 2;
13
14      printf("Before swapping:\n");
15      printf("a: %d\n", a);
16      printf("b: %d\n", b);
17
18      swap(&a, &b);
19
20      printf("After swapping:\n");
21      printf("a: %d\n", a);
22      printf("b: %d\n", b);
23
24      return 0;
25  }
```

The following is the ouput of the program's execution.

```
Before swapping:
a: 1
b: 2
```

```
After swapping:
a: 2
b: 1
```

When `swap(a,b)` is called, the memory address of `a` and `b` is passed into the function. Therefore, in line 4-7, the `x` and `y` will point to the memory of the **actual parameter** that is inserted in line 18, therefore we are doing assignments directly to the actual parameter. When passing by reference, we can't call the function with parameter that has no memory address. As an example `swapAndReturnTheSum(1,2)` cannot be done as the number 1 and 2 doesn't have memory address.

2.6 Pre-lab Assignment

1. What is the advantages and disadvantages of function?
2. Create a function with 2 arguments a and b of integer type that returns $a^2 + b^2$.
3. What are the problems that can be more easily solved with functions?

3 Recursion

Recursion refers to something that is done repeatedly Recursion in programming is when a function calls itself within its function body. As an example, take a look at the code below.

Listing 3: Factorial with a recursion

```
1    int factorial(int n) {
2    if (n==1)
3        return 1;
4    return n*factorial(n-1);
5 }
```

The factorial function calls another factorial function in line 4. Initialy, the function $factorial(n)$ is called. This function however will return $n \times factorial(n-1)$, then $factorial(n-1)$ will return $(n-1) \times factorial(n-1-1)$. Eventually it became like this:

$$\begin{aligned} factorial(n) &= n \times factorial(n-1) \\ &= n \times (n-1) \times factorial(n-2) \\ &= n \times (n-1) \times (n-2) \times \cdots \times 2 \times factorial(1) \\ &= n \times (n-1) \times (n-2) \times \cdots \times 2 \times 1 \end{aligned}$$

3.1 Pre-lab Assignment

1. What are the problems that can be more easily solved with recursion?
2. what happens if we delete the 2nd and 3rd rows in Listing 3?
3. Given a sequence of numbers 1, 5, 14, 30, 55, 91, ... etc. Create a program that implements a recursive function to determine the nth number in the pattern.