Programming Fundamentals Lab

Lab # 03

Functions

Instructor: Fariba Laiq

Email: fariba.laiq@nu.edu.pk

Course Code: CL1002

Semester Spring 2023

Department of Computer Science,

National University of Computer and Emerging Sciences FAST
Peshawar Campus

Function

- A function is a collection of statements that performs a specific task and it runs only when called.
- They let you divide complicated programs into manageable pieces.
- Functions are commonly used to break a problem down into small manageable pieces. Instead of writing one long function that contains all of the statements necessary to solve a problem, several small functions that each solve a specific part of the problem can be written. These small functions can then be executed in the desired order to solve the problem.

Advantage of functions

- By using functions, we can avoid rewriting same logic/code again and again in a program.
- We can call C functions any number of times in a program and from any place in a program.
- We can track a large C program easily when it is divided into multiple functions.

Types of Functions

There are two types of functions in C programming:

1. Library Functions:

They are the functions which are declared in the C header files such as scanf(), printf(), etc.

2. User-defined functions:

They are the functions which are created by the C programmer, so that he/she can use it many times. It reduces the complexity of a big program and optimizes the code.

Components of a Function

There are three components of a C function.

i. Function declaration

A function must be declared globally in a c program to tell the compiler about the function name, function parameters, and return type. But in case you define a function before main(), in that case you don't need to declare the function.

ii. Function call

Function can be called from anywhere in the program. The parameter list must not differ in function calling and function declaration. We must pass the same number of arguments as it is declared in the function declaration.

iii. Function definition

It contains the actual statements which are to be executed. It is the most important aspect to which the control comes when the function is called. Here, we must notice that only one value can be returned from the function.

Creating a Function in C

The syntax of creating function in c language is given below:

```
return_type function_name (data_type parameter...)
{
    // body of the function containing the code to be executed
}
```

Return Value

A C function may or may not return a value from the function. If you don't have to return any value from the function, use void for the return type. Let's see a simple example of C function that doesn't return any value from the function.

Example without return value:

```
void hello()
{
   printf("hello c");
}
```

If you want to return any value from the function, you need to use any data type such as int, long, char, etc. The return type depends on the value to be returned from the function. Let's see a simple example of C function that returns int value from the function.

Example with return value:

```
int get()
{
   return 10;
}
```

In the above example, we have to return 10 as a value, so the return type is int. If you want to return floating-point value (e.g., 10.2, 3.1, 54.5, etc), you need to use float as the return type of the method.

```
float get()
{
   return 10.2;
}
```

Now, you need to call the function, to get the value of the function.

Different ways to define a Function

A function may or may not accept any argument. It may or may not return any value. Based on these facts, there are four different aspects of function.

1. function without arguments and without return value
2. function with arguments and without return value
3. function without arguments and with return value
4. function with arguments and with return value

Now lets's first make a sampel program which reverses a 3 digit number.

- This C program takes an integer n with the value 123 and performs some mathematical operations on its digits to rearrange them in reverse order (i.e., from 123 to 321). Here's a step-by-step explanation of what the program does:
- int n = 123;: This line initializes an integer variable n with the value 123.
- int num1 = n / 100;: This line calculates the first digit (num1) by dividing n by 100. Since integer division truncates the decimal part, n / 100 results in 1.
- int num2 = (n % 100) / 10;: This line calculates the second digit (num2) by taking the remainder of n when divided by 100 (which is 23) and then dividing that result by 10. So, (n % 100) / 10 results in 2.
- int num3 = (n % 100) % 10;: This line calculates the third digit (num3) by taking the remainder of n when divided by 100 (which is 23) and then taking the remainder of that result when divided by 10. So, (n % 100) % 10 results in 3.
- int result = (num3 * 100) + (num2 * 10) + (num1);: This line calculates the final result by multiplying each digit by the appropriate power of 10 and adding them together. It essentially reverses the order of the digits. In this case, num3 (3) is multiplied by 100, num2 (2) is multiplied by 10, and num1 (1) is added as is. So, the result is 300 + 20 + 1, which is 321.
- printf("Result: %d", result);: This line prints the result to the console, which should display "Result: 321".

```
#include<stdio.h>
int main()
{
        int n=123;
        int num1=n/100;
        int num2=(n%100)/10;
        int num3=(n%100)%10;

        int result=(num3*100)+(num2*10)+(num1);
        printf("Result: %d", result);
}
```

Output: Result: 321

Now we will do the same task using functions.

1. Function without argument and return value

Example 1

```
#include<stdio.h>
void reverse()
{
	printf("Enter a no: ");
	int n;
	scanf("%d", &n);
	int num1=n/100;
	int num2=(n%100)/10;
	int num3=(n%100)%10;

	int result=(num3*100)+(num2*10)+(num1);
	printf("Result: %d", result);
}
int main()
{
	reverse(); // function call transfers the control to the function
}
```

2. Function with argument and without return value

Example

```
#include<stdio.h>
void reverse(int n) // to catch the passed argument we have to define a parameter
{
	int num1=n/100;
	int num2=(n%100)/10;
	int num3=(n%100)%10;

	int result=(num3*100)+(num2*10)+(num1);
	printf("Result: %d", result);
}
int main()
{
	printf("Enter a no: ");
	int n;
	scanf("%d", &n);
	reverse(n); // here we pass n as an argument
}
```

3. Function without argument and with return value

Example 3

```
#include<stdio.h>
int reverse()
{
	printf("Enter a no: ");
	int n;
	scanf("%d", &n);
	int num1=n/100;
	int num2=(n%100)/10;
	int num3=(n%100)%10;
	int result=(num3*100)+(num2*10)+(num1);
	return result; // it will return the result back to main, specifically to the point of function call
}
int main()
{
	int result=reverse(); // to catch the returned value we need to store it in a variable
	printf("Result: %d", result);
}
```

4. Function with argument and with return value

Example 4

```
#include<stdio.h>
int reverse(int n)
{
        int num1=n/100;
        int num2=(n%100)/10;
        int num3=(n%100)%10;
        int result=(num3*100)+(num2*10)+(num1);
        return result;
}
int main()
{
        printf("Enter a no: ");
        int n;
        scanf("%d", &n);
        int result=reverse(n);
        printf("Result: %d", result);
}
```

Python Tutor – A great tool to visualize your code.

Just remove the lines where you take input from user and write hard coded values as this tool does not work with scanf(). Visualize your code to keep a track and flow of every variable and line. It helps you greatly to solve your bugs.

https://pythontutor.com/render.html#mode=display

Python Tutor: Visualize code in Python, JavaScript, C, C++, and Java

C (C17 + GNU extensions, gcc 9.3)
known limitations

```
 1  #include<stdio.h>
 2  int reverse(int n)
 3  {
 4          int num1=n/100;
 5          int num2=(n%100)/10;
 6          int num3=(n%100)%10;
 7          int result=(num3*100)+(num2*10)+(num1);
 8          return result;
 9  }
10  int main()
11  {
12          int n=123;
13          int result=reverse(n);
14          printf("Result: %d", result);
15  }
```

Edit this code

line that just executed
next line to execute

<< First | < Prev | Next > | Last >>

Step 10 of 11

Print output (drag lower right corner to resize)

Stack Heap

main

n	int 123
result	int ?

reverse

n	int 123
num1	int 1
num2	int 2
num3	int 3
result	int 321

Note: ? refers to an uninitialized value