Programming for Problem Solving

(ESCG101)

Module 4

Module IV: Function

- Functions (including using built-in libraries)
- Parameter passing in functions, call by value.
- Passing arrays to functions: the idea of call.

Function

- A function is a group of statements that together performs a task
- Every C program has at least one functions, which is main( )
- Additional functions can be defined as per the requirement

Importance of function in C programming language:

Function refers to a group of statements to perform a specific task. It is useful in any programming language, especially in C programming because it not only provides large sets of built-in functions but also user-defined functions for executing many tasks. Here are other reasons why it is important:

• It allows it to remove big programs\' complexity by breaking them down into small functions.
• It enhances the readability of programs so it can easily understand the codes.
• The top-down execution in C lets it learn the program flow and control every line.
• It can also create itr header file and reuse it in other programs.
• It can test individual functions and make codes prune to fewer errors.

Types of Function

- Two types
  - Pre-defined -- these are the inbuilt functions of C library like main(), printf(), scanf() etc
  - User-defined -- these are defined by the user

Library functions or built-in functions are functions : are pre-defined by the programming language. It can use those functions for different purposes. C language is enriched with hundreds of library functions. The main() is also a built-in function responsible for the code program execution. Some popular library functions of header files \"math.h\" pow() - It calculates the power of a number, sin() - It calculates the sine of a number, cos() - It calculates the cosine of a number, exp() - It calculates the exponent of a number, sqrt() - It calculates the square root of a number.

➢**Can a C program be compiled or executed in the absence of a main()?**

The program will be compiled but will not be executed. To execute any C program, main() is required.

Objective of the main () function in C:

The main () function in C to the inlet to the C program. It is the entry point where the process of execution of the program starts. When the execution of the C program initiates, the control of the program is directed towards the main () function.

It is mandatory that every C program has a main () function. Although it is the function that indicates the programming process, it is not the first function to be executed.

In C, both constants and variables are widely used while designing a program. The major difference between variables and constants is that variables can alter their assigned value at any point of the program. In contrast, the value of the constant remains unaltered during the entire program. The value of the constant is locked during the execution of the program.

How does a function operate?

- Codes can be divided into separate parts
- Those parts can be used to create functions
- Example:

```
int main()  // main is pre defined
{                                  function
    //statements
}
void show()  // show is user defined function
 {
  // statements
}
```

Syntax

```
return_type function_name(arguments)
{
        statements;
}
```

-here, return type can be void, int, float, char.

Function

- **Calling function-** the function who calls other function
- **Called fuction-** the function who get called by other function
- **Actual parameters-** arguments which are passed to called function
- **Formal parameters-** arguments of the called function

Example

```
int main()                                    calling function
{
int a=5, b=2;                                      called function

addition(a,b);    // a & b are actual parameters
}

    void addition(int x, int y) // x & y are formal
    {
     int sum=x+y;
    }
```

Function prototype

- Function prototype is needed when user defined function is defined after main function
- It is just the declaration of the function with any body
- Example :

```
void show()
{
   // statements
}
// here prototype will be void show(); defined before main
   function
```

Local Variable: A local variable is a type of variable that we declare inside a block or a function.

Global Variable: A global variable is one that is defined outside the scope of all functions. Because global variables have a global scope, they can be accessed and modified by any function, structure, or scope in C.

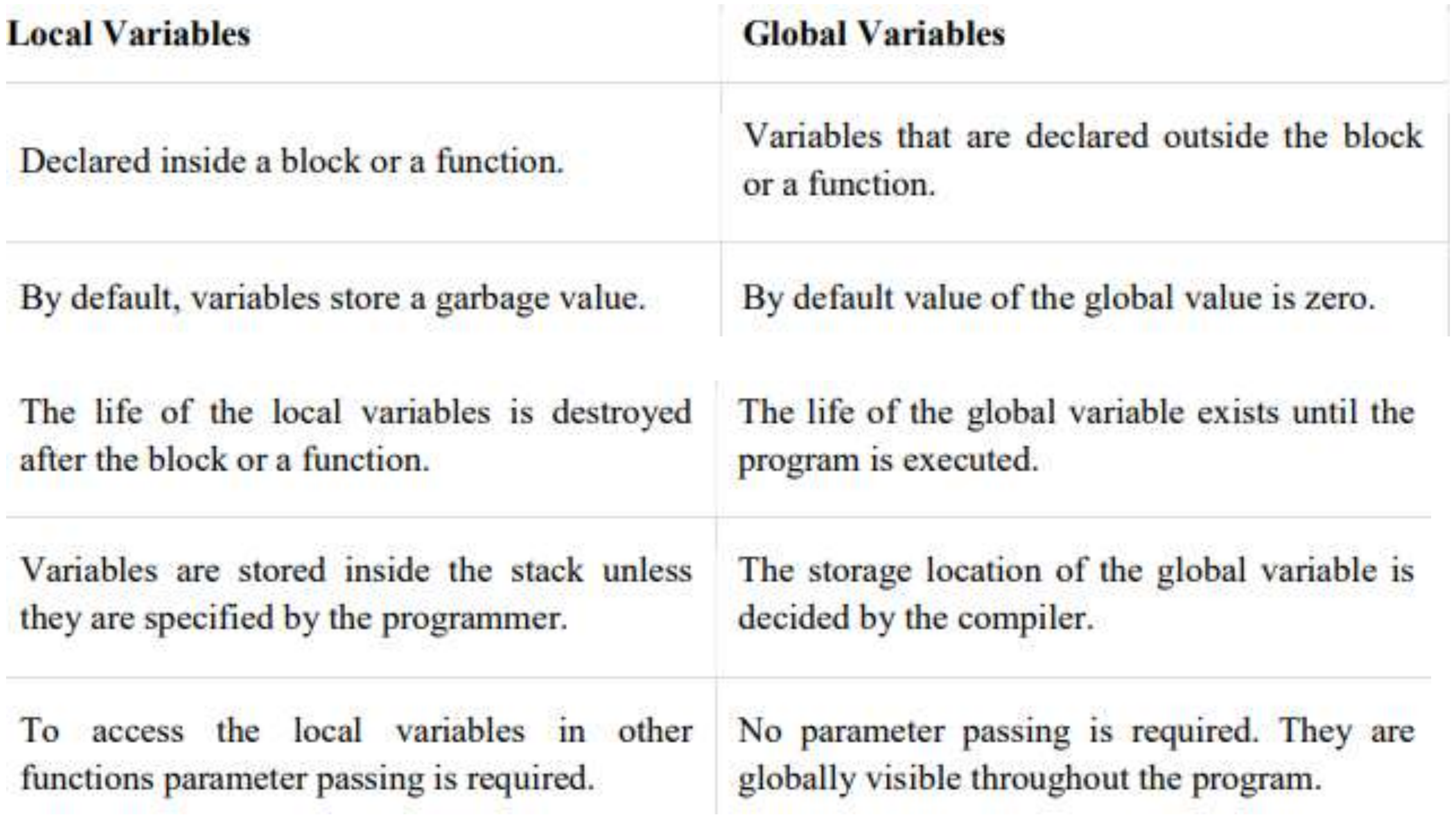
Static Variable: A static variable possesses the property of preserving its actual value even after it is out of its scope.

- In C it is always the most local variable that gets preference.
- Same variable name can be declared to the variables with different variable scopes as the following example.

```
int var;
void function()
{
    int variable;
 }
int main()
{
      int variable;
}
```

Difference between the local and global variables in C :

Local variables are declared inside a block or function but global variables are declared outside the block or function to be accessed globally.



Local Variables	Global Variables
Declared inside a block or a function.	Variables that are declared outside the block or a function.
By default, variables store a garbage value.	By default value of the global value is zero.
The life of the local variables is destroyed after the block or a function.	The life of the global variable exists until the program is executed.
Variables are stored inside the stack unless they are specified by the programmer.	The storage location of the global variable is decided by the compiler.
To access the local variables in other functions parameter passing is required.	No parameter passing is required. They are globally visible throughout the program.

Differentiate between getch() and getche():

Both the functions are designed to read characters from the keyboard and the only difference is that getch(): reads characters from the keyboard but it does not use any buffers. Hence, data is not displayed on the screen. getche(): reads characters from the keyboard and it uses a buffer. Hence, data is displayed on the screen.

toupper() function with an example: toupper() is a function designed to convert lowercase words/characters into upper case.

```
#include<stdio.h>
#include<ctype.h>
int main()
{
    char c;
    c=a;
    printf(\"%c after conversions %c\", c, toupper(c));
    return 0;
}
Output: A
```

rand() function: Random numbers in C Language can be generated as follows:

```
#include<stdio.h>
#include<stdlib.h>
 int main()
{
   int a,b;
   for(a=1;a<=10;a++)
   {
      b=rand();
      printf("%d\n",b);
   }
   return 0;
}
```

Output:

```
1804289383
846930886
1681692777
1714636915
1957747793
424238335
719885386
1649760492
596516649
1189641421
```

Limitations of scanf() and how can it be avoided: The Limitations of scanf() are as follows:

- scanf() cannot work with the string of characters.
- It is not possible to enter a multiword string into a single variable using scanf().

To avoid this the gets( ) function is used. It gets a string from the keyboard and is terminated when enter key is pressed. Here the spaces and tabs are acceptable as part of the input string.

Difference between macros and functions C programming language:

Macro : Macro is a small chunk of code which gets replaced whenever respective macros have been called. It is called a preprocessor directive, and it requires pre-processing. It does not require any type-checking. The increased number of macros increases the size of the whole program. Macro execution is faster than function execution. Use macro when it need to execute small codes multiple times.

Function : A function is a group of one or more statements to perform a specific job. It could be predefined or user-defined, and it needs compilation. It requires type-checking. The function length does not impact the size of the program. The function execution is slower than the macro execution. Use functions when executing large codes and different tasks in a program.

String functions in the C language:

C provides many string functions in its standard library header file- ‘string.h’. The most commonly used string functions are:

strlen(string_source): Returns length of the string_source

strcpy(string_destination, string_source): Copies string_source into string_destination.

strcat(string_destination, string_source): Joins both the strings and stores the output in string_destination.

strcmp(string_destination, string_source): Compares both strings, if found equal returns 0(zero).

strrev(string_source): Reverses the string

strlwr(string_source): Returns the string into lowercase

strupr(string_source) Returns the string into uppercase

Function prototype with an example:

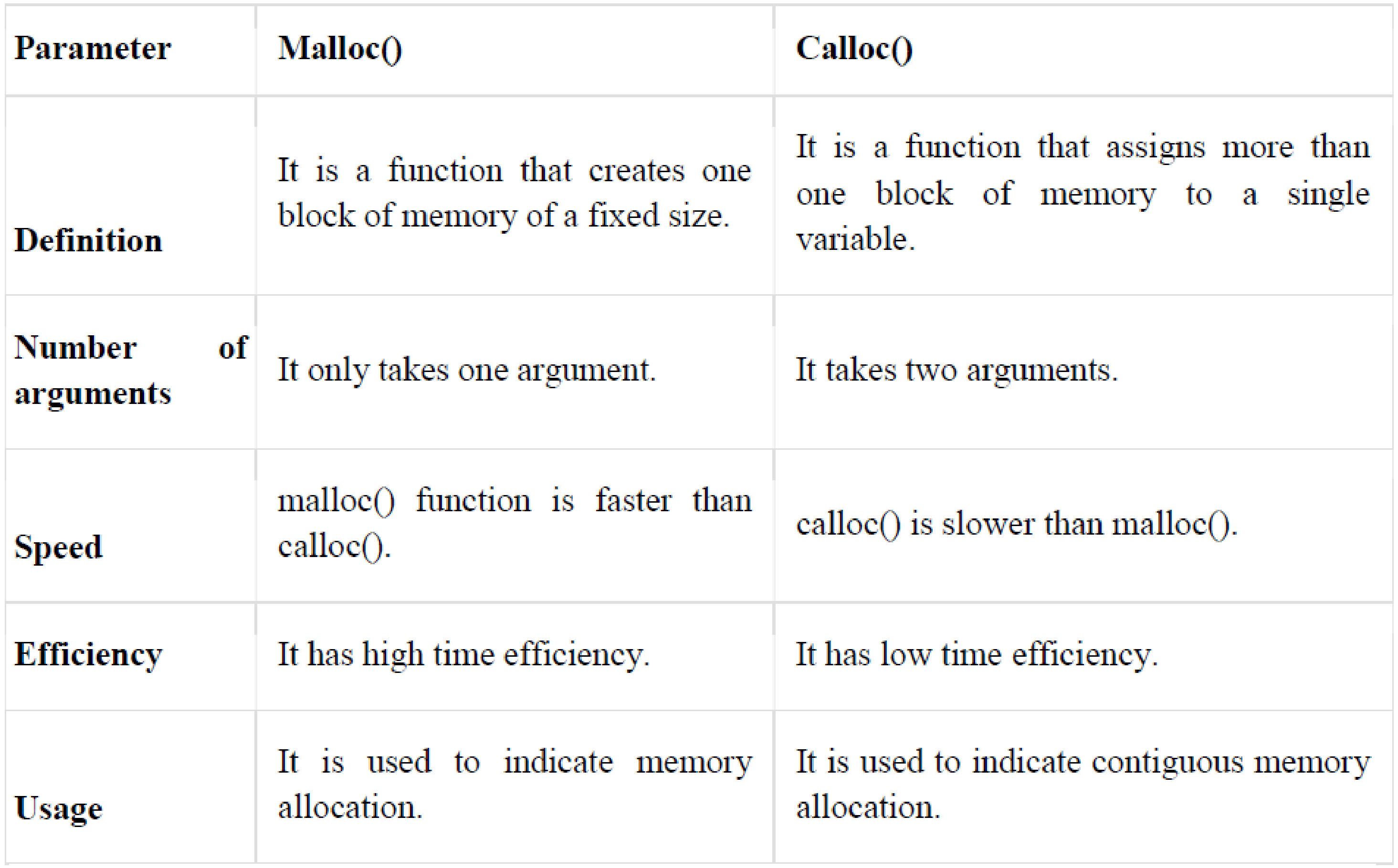
Function prototype of prototype function refers to the declaration of any function with the following information:

- Name of function
- Function return type
- Number of arguments and their data type.

The function prototype provides the assurance that the function call matches the return type and returns the correct number of arguments of the called function. Simply put, it helps prevent the user from making mistakes while using multiple functions in the program. Syntax: return_type function_name(datatype argument1, datatype argument 2, …);

Example: int sum(int x, int y); This line is declaring the function \"sum\" which is taking two integer arguments. The function will also return the integer.

Difference between malloc() and calloc() in the C programming language:



Parameter	Malloc()	Calloc()
Definition	It is a function that creates one block of memory of a fixed size.	It is a function that assigns more than one block of memory to a single variable.
Number of arguments	It only takes one argument.	It takes two arguments.
Speed	malloc() function is faster than calloc().	calloc() is slower than malloc().
Efficiency	It has high time efficiency.	It has low time efficiency.
Usage	It is used to indicate memory allocation.	It is used to indicate contiguous memory allocation.

C program to convert number to string using sprintf():

```
#include <stdio.h>
#include <string.h>
// Driver code
int main()
{
    char res[20];
    float a = 32.23;
    sprintf(res, "%f", a);
    printf("\nThe string for the num is %s", res);
    return 0;
}
```

Output:

```
The string for the num is 32.230000
```

C program to check whether a string is palindrome or not:

A string is said to be palindrome if it remains the same on reading from both ends. It means that when you reverse a given string, it should be the same as the original string.

```
#include <stdio.h>
#include <string.h>
void Palindrome(char s[]);
int main()
{
   Palindrome("abba");
   return 0;
}
void Palindrome(char s[])
{
/* start will start from 0th index and end will start from length-1 */
   int start = 0;
   int end = strlen(s) - 1;
/* Comparing characters until they are same */
   while (end > start)
   {
      if (s[start++] != s[end--])
      {
         printf("%s is not a Palindrome \n", s);
      }
   }
  printf("%s is a Palindrome \n", s);
}
```

Output:

```
abba is a Palindrome
```

C program to find out the factorial of a number using function:

```
# include<stdio.h>
long int factorial(int n);
int main()
{
   int num;
   printf("Enter a number :");
   scanf("%d", &num);
   if (num<0)
      printf("No factorial of negative number\n");
   else
      printf("Factorial of %d is %ld\n", num, factorial(num));
   return 0;
}
long int factorial(int n)
{
   int i;
   long int fact=1;
   if (n==0)
      return 1;
   for (i=n; i>1; i--)
      fact*=i;
   return fact;
}
```

Output:

```
Enter a number: 5
Factorial of 5 is 120
```

Thank You