Pemrograman Dasar

Week 4 : *Function*

UMM
UNIVERSITAS MUHAMMADIYAH MALANG

OUTLINES

1. Scope Rules
2. Function
3. Function Arguments

1. Scope Rules

A scope in any programming is a **region of the program where a defined variable** can have its existence and beyond that variable it cannot be accessed.

Here are **three places** where variables can be declared in C programming language:

1. Inside a function or a block which is called local variables.
2. Outside of all functions which is called global variables.
3. In the definition of function parameters which are called formal parameters.

Local Variables

Variables that are declared **inside a function or block** are called local variables. They can be used only by statements that are inside that function or block of code. **Local variables are not known to functions outside their own**.

Local Variable [PROGRAM]

ProgDas4-1.c

```
#include <stdio.h>
int main () {
  /* local variable declaration */
  int a, b;
  int c;
  /* actual initialization */
  a = 10;
  b = 20;
  c = a + b;

  printf ("value of a = %d, b = %d and c = %d\n", a, b, c);
  return 0;
}
```

Global Variables

Global variables are defined **outside a function**, usually on top of the program. Global variables hold their values throughout the lifetime of your program and they **can be accessed inside any of the functions defined for the program**.

That is, a global variable is available for use throughout your entire program after its declaration.

Global Variable [PROGRAM]

ProgDas4-2.c

```
#include <stdio.h>
int g;

int main () {
  int a, b;
  /* actual initialization */
  a = 10;
  b = 20;
  g = a + b;

  printf ("value of a = %d, b = %d and g = %d\n", a, b, g);
  return 0;
}
```

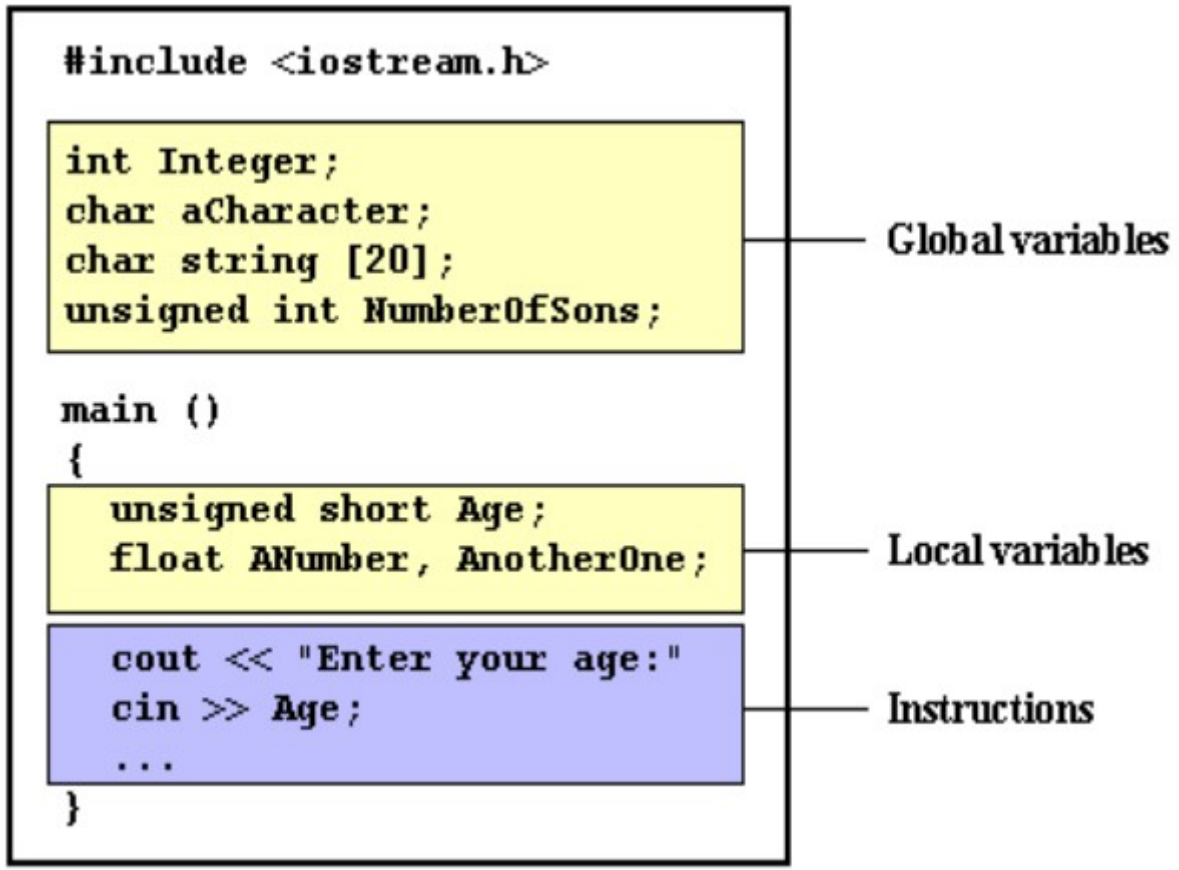
Note: A program can have same name for local and global variables but the value of local variable inside a function will take preference.

ProgDas4-3.c

```
#include <stdio.h>
int g = 20;

int main () {
   /* local variable declaration */
   int g = 10;

   printf ("value of g = %d\n",  g);
   return 0;
}
```


#include <iostream.h>
int Integer;
char aCharacter;
char string [20];
unsigned int NumberOfSons;
main ()
{
unsigned short Age;
float ANumber, AnotherOne;
cout << "Enter your age:"
cin >> Age;
...
}
Global variables
Local variables
Instructions

Initializing Local and Global Variables

When a local variable is defined, it is not initialized by the system, you must initialize it yourself. Global variables are initialized automatically by the system

Data Type	Initial Default Value
int	0
char	'\0'
float	0
double	0
pointer	NULL

2. Function

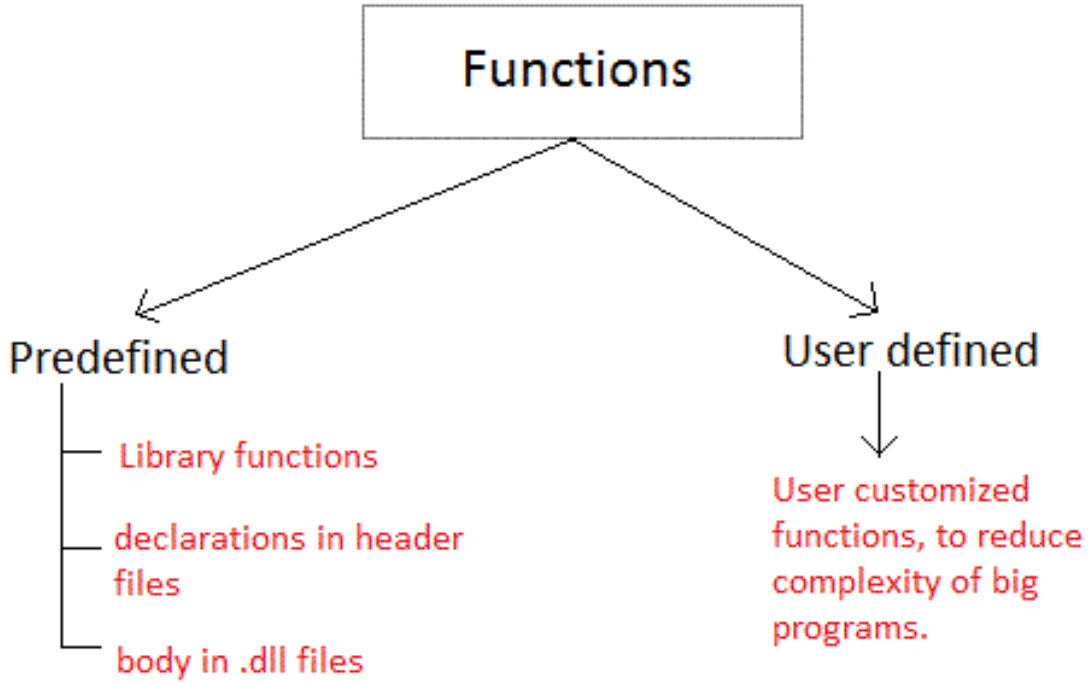
A function is a group of statements that together perform a task. Every C program has at least one function, which is `main()`, and all the most trivial programs can define additional functions.

You can divide up your code into separate functions. How you divide up your code among different functions is up to you, but **logically the division is such that each function performs a specific task**.

These functions defined by the user are also know as **User-defined Functions**

C functions can be classified into two categories,

1. Predefined
2. User-defined functions



Predefined (Library functions) are those functions which are already defined in C library, example printf(), scanf(), strcat() etc. You just need to include appropriate header files to use these functions. These are already declared and defined in C libraries.

A User-defined functions on the other hand, are those functions which are defined by the user at the time of writing program. These functions are made for code reusability and for saving time and space.

Predefined Function

String Operation Function (stored in the file header "string.h")

`strcpy ()`

Function to copy an original string to the destination string variable.

General form: `strcpy (var_tujuan, string_asal);`

`strlen ()`

function to obtain the number of characters from a string.

General form: `strlen (string);`

ProgDas4-4.2.c

```
#include <stdio.h>
#include <conio.h>
#include <string.h>
void main()
{
  char nama[25];
  strcpy(nama, "Achmad Solichin");
  printf("Nama : %s\n", nama);
  printf("Banyaknya karakter nama Anda adalah : %i", strlen(nama));
  getch();
}
```

Advantages of user-defined function

1. The program will be easier to understand, maintain and debug.
2. Reusable codes that can be used in other programs
3. A large program can be divided into smaller modules. Hence, a large project can be divided among many programmers.

A function **declaration** tells the compiler about a function's name, return type, and parameters. A function **definition** provides the actual body of the function.

```
return_type function_name(parameter_list)
{
   body of the function
}
```

A function definition in C programming consists of a **function header** and a **function body**

Remember, function name is an identifier and should be unique.

Here are all the **parts of a function**:

Return Type – A function may return a value. The return_type is the data type of the value the function returns. Some functions perform the desired operations without returning a value. In this case, the return_type is the keyword void.

Function Name – This is the actual name of the function. The function name and the parameter list together constitute the function signature.

Parameters – A parameter is like a placeholder. When a function is invoked, you pass a value to the parameter. This value is referred to as actual parameter or argument. The parameter list refers to the type, order, and number of the parameters of a function. Parameters are optional; that is, a function may contain no parameters.

Function Body – The function body contains a collection of statements that define what the function does.

```
#include <stdio.h>
void functionName()
{
    … .. …
    … .. …
}

int main()
{
    … .. …
    … .. …

    functionName();

    … .. …
    … .. …
}
```

When the compiler encounters `functionName();` inside the `main()` function, control of the program jumps to `void functionName()`

And, the compiler starts executing the codes inside the user-defined function.

The control of the program jumps to statement next to `functionName();` once all the codes inside the function definition are executed.

Example

Given below is the source code for a function called max(). This function takes two parameters num1 and num2 and returns the maximum value between the two –

```
/* function returning the max between two numbers */
int max(int num1, int num2) {
   /* local variable declaration */
   int result;
   if (num1 > num2)
      result = num1;
   else
      result = num2;
   return result;
}
```

Calling a Function

While creating a C function, you give a definition of what the function has to do. To use a function, you will have to call that function to perform the defined task.

When a program calls a function, the program control is transferred to the called function. A called function performs a defined task and when its return statement is executed or when its function-ending closing brace is reached, it returns the program control back to the main program.

To call a function, you simply need to pass the required parameters along with the function name, and if the function returns a value, then you can store the returned value.

ProgDas4-5.c

As mentioned earlier, C allow programmers to define functions. Such functions created by the user are called user-defined functions.

Depending upon the complexity and requirement of the program, you can create as many user-defined functions as you want.

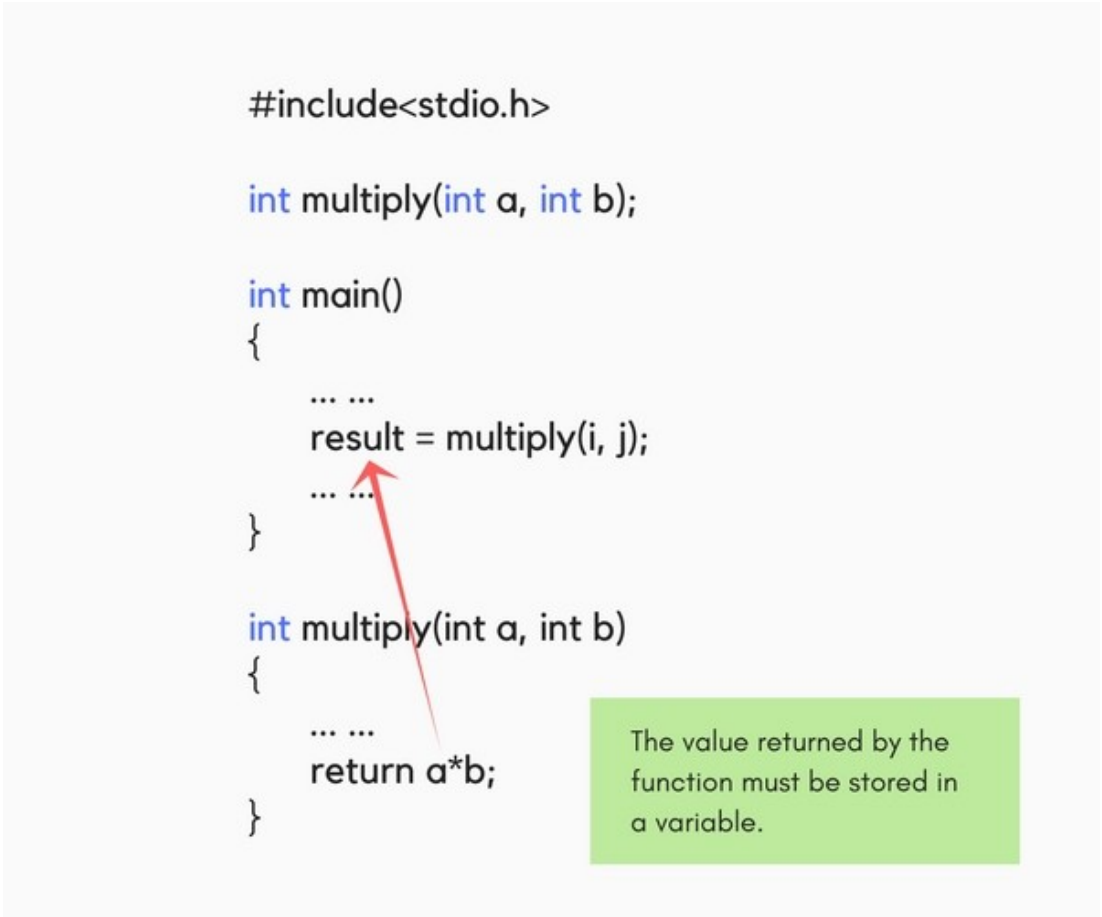
The execution of a C program begins from the `main()` function.

Returning a Value from Function

ProgDas4-5.2.c

A function may or may not return a result. But if it does, we must use the return statement to output the result. return statement also ends the function execution, hence it must be the last statement of any function. **If you write any statement after the return statement, it won't be executed**.

Returning a Value from Function



Important points about functions in C (1)

- Every function has a return type. If a function doesn't return any value, then void is used as return type. Moreover if the return type of the function is void ,we still can use return statement in the body of function definition by not specifying any constant,variable,etc. with it ,by only mentioning the 'return;' statement which would symbolise the termination of the function as shown below:

```
void function name(int a)
{
.......  //Function Body
return;  //Function execution would get terminated
}
```

Important points about functions in C (2)

- In C, functions can return any type except arrays and functions.

- Empty parameter list in C mean that the parameter list is not specified and function can be called with any parameters. In C, it is not a good idea to declare a function like fun(). To declare a function that can only be called without any parameter, we should use “void fun(void)”.

- If the return data type of a function is “void”, then, it can’t return any values to the calling function.
- If the return data type of the function is other than void such as “int, float, double etc”, then, it can return values to the calling function.

3. Function Arguments

Function Arguments

If a function is to use arguments, it must declare variables that accept the values of the arguments. These variables are called the formal parameters of the function.

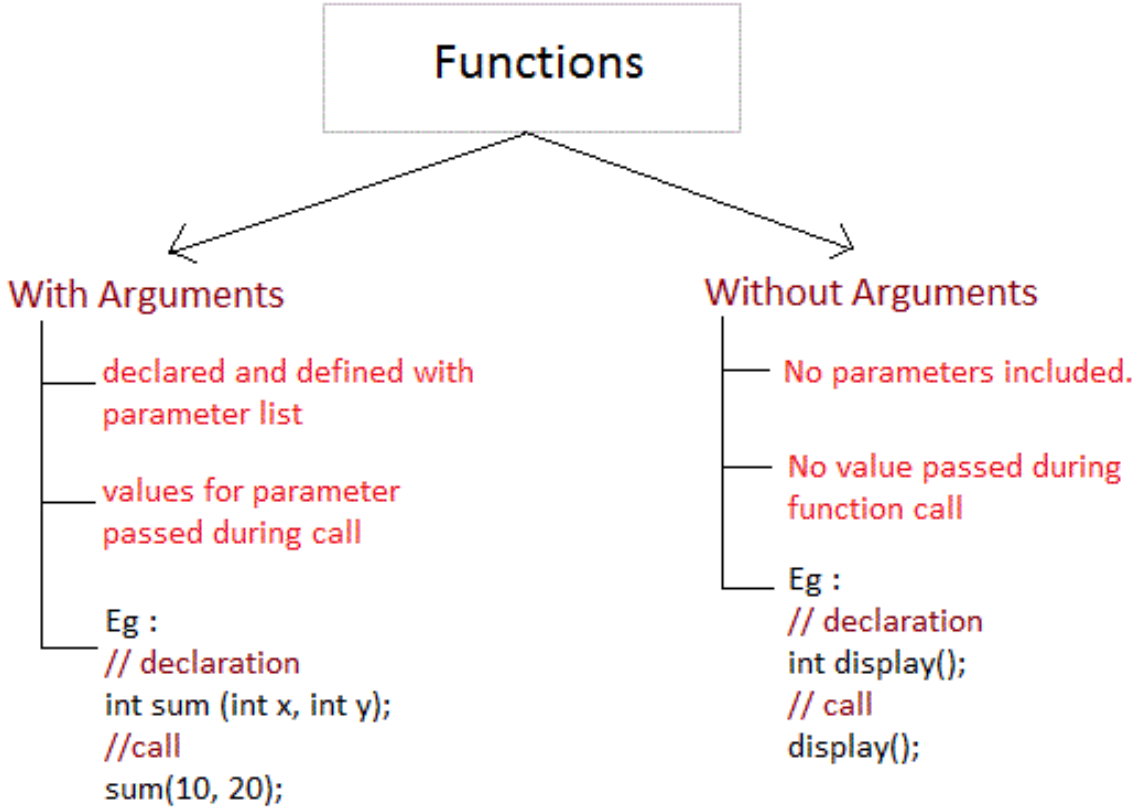
Formal parameters behave like other local variables inside the function and are created upon entry into the function and destroyed upon exit.

While calling a function, there are two ways in which arguments can be passed to a function

Passing Arguments to a function

Arguments are the values specified during the function call, for which the formal parameters are declared while defining the function.

It is possible to have a function with parameters but no return type. It is not necessary, that if a function accepts parameter(s), it must return a result too.



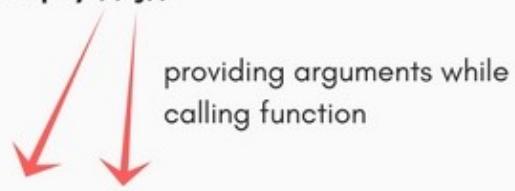
Passing Arguments to a function

```
#include<stdio.h>

int multiply(int a, int b);

int main()
{
    ... ...
    result = multiply(i, j);
    ... ...
}

int multiply(int a, int b)
{
    ... ...
}
```



Actual parameters: The parameters that appear in function calls.
Formal parameters: The parameters that appear in function declarations.

Formal Parameter

```
float tambah(float x, float y) //formal parameter
{
   return (a+b);
}
```

Actual Parameter

```
void main()
{
   …
   c = tambah(a, b);   //actual parameter
   …
}
```

Call Type & Description

1. Call by value **ProgDas4-5.3.c**

This method copies the actual value of an argument into the formal parameter of the function. **In this case, changes made to the parameter inside the function have no effect on the argument.**

ProgDas4-5.4.c

2. Call by reference

This method copies the address of an argument into the formal parameter. Inside the function, the address is used to access the actual argument used in the call. **This means that changes made to the parameter affect the argument.**

1. Function with **no argument** and **no return value**

ProgDas4-6.c

When a function has no arguments, it does not receive any data from the calling function. Similarly when it does not return a value, the calling function does not receive any data from the called function.

Syntax :

```
Function declaration : void function();
Function call : function();
Function definition :
                    void function()
                    {
                      statements;
                    }
```

2. Function with **arguments** but **no return value**

ProgDas4-7.c

When a function has arguments, it receive any data from the calling function but it returns no values.

Syntax :

```
Function declaration : void function ( int );
Function call : function( x );
Function definition:
            void function( int x )
            {
              statements;
            }
```

3. Function with **no arguments** but **returns a value**

There could be occasions where we may need to design functions that may not take any arguments but returns a value to the calling function. A example for this is getchar function it has no parameters but it returns an integer an integer type data that represents a character.

Syntax :

```
Function declaration : int function();
Function call : function();
Function definition :
                int function()
                {
                    statements;
                     return x;
                 }
```

4. Function with **arguments** and **return value**

Syntax :

```
Function declaration : int function ( int );
Function call : function( x );
Function definition:
            int function( int x )
            {
              statements;
              return x;
            }
```

Do You Know How Many Values Can Be Return From C Functions?

- Always, Only one value can be returned from a function.
- If you try to return more than one values from a function, only one value will be returned that appears at the right most place of the return statement.
- For example, if you use “return a,b,c” in your function, value for c only will be returned and values a, b won’t be returned to the program.
- In case, if you want to return more than one values, pointers can be used to directly change the values in address instead of returning those values to the function.

EXERCISE

Temperature Conversion

Centigrade (Celcius)

Boiling point of water: 100 C

Freezing point: 0 C

From 0 degrees Celsius to 100 degrees Celsius divided into 100 scales.

Reamur

Boiling point of water: 80 R

The freezing point: 0 R

From 0 degrees of Reamur to 80 degrees of Reamur divided into 80 scales.

Fahrenheit

Boiling point of water: 212 F

Freezing point: 32 F

From 32 degrees Fahrenheit to 212 degrees Fahrenheit is divided into 180 scales.

Kelvin

Boiling point of water: 373 K

The freezing point: 273 K

From 273 Kelvin to 373 Kelvin divided into 100 scales.

The conversion of temperatures from Celsius (C) to Reamur (R), Fahrenheit (F), and Kelvin (K) are:

$R = (4/5)\ C$

$F = (9/5)\ C + 32$

$K = C + 273$

Temperature conversion from Reamur (R) to Celsius (C), Fahrenheit (F), and Kelvin (K) are:

$C = (5/4)\ R$

$F = (9/4)\ R + 32$

$K = C + 273 = (5/4)\ R + 273$

The conversion of temperatures from Fahrenheit (F) to Celsius (C), Reamur (R), and Kelvin (K) are:

$C = 5/9\ (F-32)$

$R = 4/9\ (F-32)$

$K = 5/9\ (F-32) + 273$

The temperature conversion from Kelvin (K) to Celsius (C), Reamur (R), Fahrenheit (F) is:

$C = K - 273$

$R = 4/5\ (K-273)$

$F = 9/5\ (K-273) + 32$

Create a program to convert temperature from centigrade to Fahrenheit, Kelvin, Reamur using 4 functions

THANKYOU