PROGRAMMING

Subject Title: Programming in 'C'
Subject Code: CAM203-1C

Unit-1 User-Defined Functions

Prepared by Ankit Rami(AR)

Any Query Contact – 8460467193

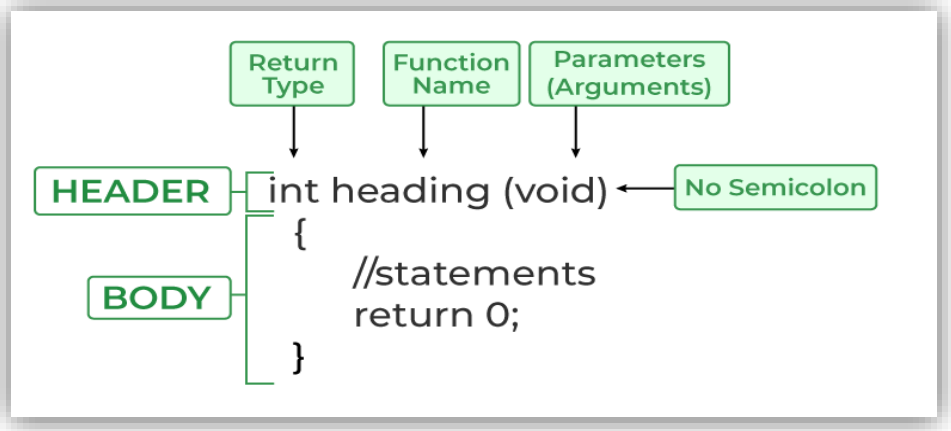
Email – ankitramiblog@gmail.com

📢 Introduction for UDF(User Defined Functions)

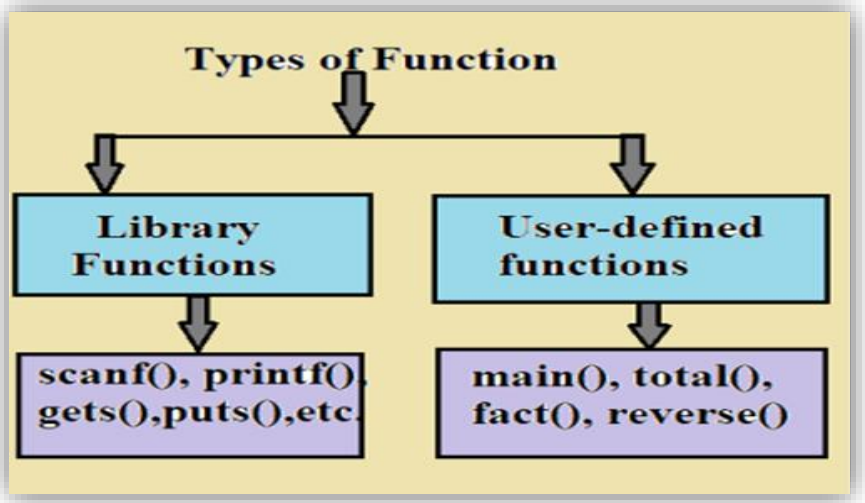
- ***User-defined functions*** are a powerful feature of C Programming
- A ***User-defined functions*** is a block of code that performs a specific task.
- In Side of Functions Developer can set of statements that are combined to perform a specific task.
- C Programming allows you to define functions according to your need. These functions are known as user-defined functions.
- These functions can be called anywhere in the program, making the code more modular and easier to read.

Introduction for UDF(User Defined Functions)

- ***User-defined functions*** Syntax



- ***Types of Functions***

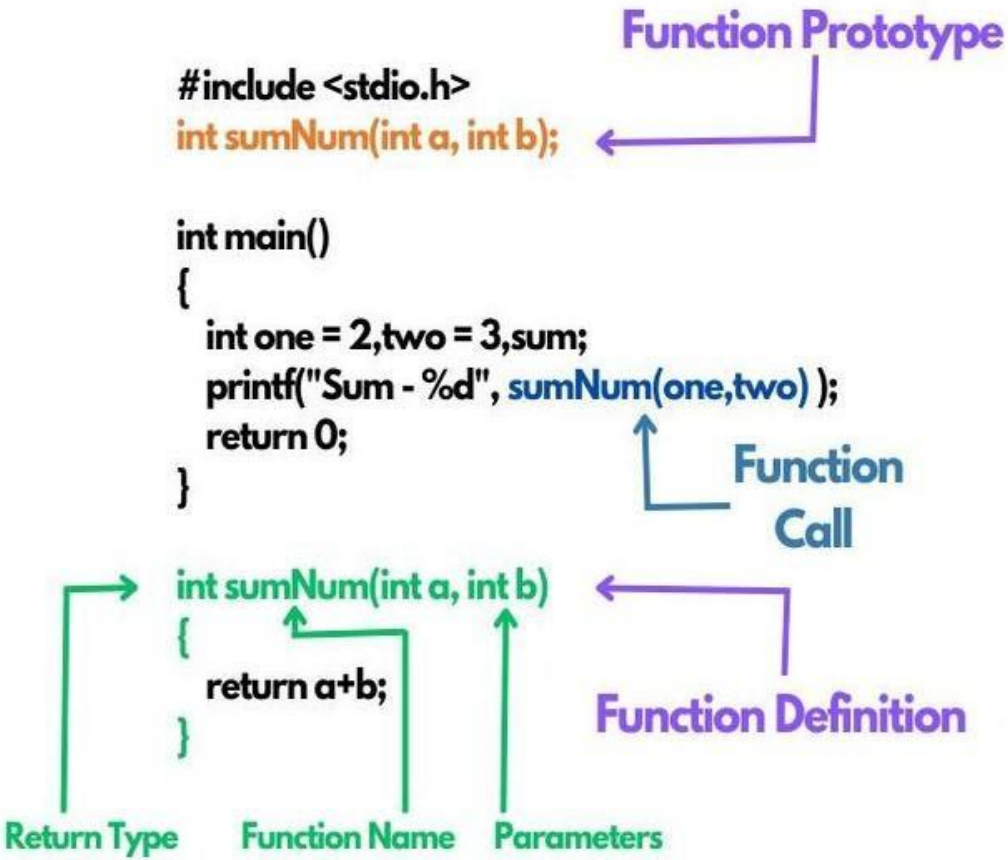


Need for *User-defined functions*

- ✓ **Reduction in Program Size -** This avoids writing of same code again and again reducing program size.
- ✓ **Reducing Complexity of Program:** Complex program can be decomposed into small sub-programs or user defined functions.
- ✓ **Easy to Debug and Maintain :** During debugging it is very easy to locate and isolate faulty functions. It is also easy to maintain program that uses user defined functions.
- ✓ **Readability of Program:** Since while using user defined function, a complex problem is divided in to different sub-programs with clear objective and interface which makes easy to understand the logic behind the program.
- ✓ **Code Reusability:** Once user defined function is implemented it can be called or used as many times as required which reduces code repeatability and increases code reusability.

Elements of UDF

- Function Declaration
- Function Definition
- Function Call



Elements of UDF

- **Function Declaration**
- **Syntax -** void sum(); int ar(); float ar(); etc…
- A function declaration is also known as a function prototype. If the function definition is mentioned before the main function, then the function prototype is not necessary to mention.we can eliminate the function declaration statement.
- The function declaration statement must always match the function name and the number of arguments passed with the function definition.
- From the given example, function name, sum, and the return type void must match both the statements, function declaration, and function definition. A function declaration statement contains the return type, function name (list of arguments if required), and semicolon at the end of the statement.

📣 Elements of UDF

- **Function Definition**
- **Syntax –**

```
void sum()   //function definition
{
int a=5, b=10,c;
c=a+b;
printf("addition ans is: %d",c);
}
```

- The body of function definition consists of a set of statements to perform a specific operation or task. The compiler will return to the function definition whenever a function is called and execute those statements.
- According to the given example, the function's body consists of statements to perform the addition operation. Variables a and b hold the values 5 and 10, respectively, and variable c will have the result value of 5+10 =15.

📣 Elements of UDF

- **Function Call**
- **Syntax –** sum();
- Once defining a function definition, there must be a function call in the program. Suppose a compiler encounters a function call anywhere in the program. It will transfer its control to the function definition and execute a set of statements inside the body of a function. A function can be called n number of times in a program.
- Considering the given example, sum(); is the function call. Since no arguments are passed in the function declaration and function definition, we don’t provide them in the function call.

Simple Example of UDF

D:\Programming-C AR\UDF1.c - [Executing] - Dev-C++ 5.11

File Edit Search View Project Execute Tools AStyle Window Help

TDM-GCC 4.9.2 64-bit Release

(globals)

Project Classes Debug | UDF1.c

```c
#include<stdio.h>
void sum();
int main()
{
    sum();
}
void sum()
{
    int a=5;
    int b=10;
    int c=a+b;
    printf("Addition and is: %d",c);
}
//Learn programming with Ankit Rami
```

Select D:\Programming-C AR\UDF1.exe

```
Addition and is: 15
--------------------------------
Process exited after 0.005967 seconds with return value 19
Press any key to continue . . .
```

Compiler | Resources | Compile Log | Debug | Find Results | Close

Abort Compilation

Shorten compiler paths

```
Compilation results...
--------
- Errors: 0
- Warnings: 0
- Output Filename: D:\Programming-C AR\UDF1.exe
- Output Size: 128.44921875 KiB
- Compilation Time: 0.22s
```

Line: 14 Col: 36 Sel: 0 Lines: 14 Length: 200 Insert Done parsing in 0 seconds

Type here to search | 21°C Smoke | ENG | 09:38 16-02-2024

📣 Category of User-Defined Functions in C

- ✓ Function with no argument and no returning value
- ✓ Function with argument and no returning value
- ✓ Function with no argument with returning value
- ✓ Function with argument with returning value

Reference Link for Learning

- ❖ https://computenepal.com/4-types-of-user-defined-functions-in-c/
- ❖ https://www.simplilearn.com/tutorials/c-tutorial/learn-user-defined-function-in-c#:~:text=From%20the%20given%20example%2C%20function,the%20end%20of%20the%20statement.
- ❖ https://www.javatpoint.com/user-defined-function-in-c#:~:text=User%2Ddefined%20functions%20can%20be,can%20be%20any%20valid%20identifier.

📣 No Argument and No Returning Value in C

- ✓ In this method, we don't pass any arguments and mention void as a function return type, so a function will not return any value. We declare a few variables in the function definition and perform certain operations on them.

```c
#include <stdio.h>
void sum(); ////not passing any parameters
int main()
{
  sum();
  return 0;

}
void sum()

{
  int a=5,b=10,c;
  c=a*b;
  printf("sum of two numbers=%d",c);
  //no return any value in function
}
//Learn programming with Ankit Rami
```

```
sum of two numbers=50
--------------------------------
Process exited after 0.00486 seconds with return value 0
Press any key to continue . . .
```

```
Compilation results...
--------
- Errors: 0
- Warnings: 0
- Output Filename: D:\Programming-C AR\noargumentandnoreturningvalue.exe
- Output Size: 128.482421875 KiB
- Compilation Time: 0.17s
```

With Argument and No Returning Value in C

- ✓ In this function call method, with an argument and no returning value, we pass arguments to the function, but the function doesn't return a value.
- ✓ In this program's arguments passed to the function are a and b of type integer. And it is also necessary to give the arguments in the function call.

```c
#include <stdio.h>
void sum(int a,int b); //passing int argument(parameters)
int main()
{
  sum(10,5);//assign int argument(parameters)
  return 0;

}
void sum(int a,int b)
{
  int c;
  c=a+b;
  printf("sum of two numbers=%d",c);
  //no return any value in function
}
//Learn programming with Ankit Rami
```

```
sum of two numbers=15
--------------------------------
Process exited after 0.009033 seconds with return value 0
Press any key to continue . . .
```

📣 No Argument and With Returning Value in C

- ✓ In this function call method, we don't pass any arguments in the function but mention a return type. A return type can be any data type; by default, the return type is int, and the value returned by a function is an integer value.

```c
#include <stdio.h>
int sum(); //not passing any parameters
int main()
{
   printf("sum of two numbers=%d",sum());
   return 0;

}
int sum()
{
   int a=5,b=10,c;
   c=a+b;
   return(c);//function passing return value
}
//Learn programming with Ankit Rami
```

```
sum of two numbers=15
--------------------------------
Process exited after 0.00617 seconds with return value 0
Press any key to continue . . .
```

With Argument and With Returning Value in C

✓ This method passes arguments to the function, which will also return a value.

```c
#include <stdio.h>
int sum(int a,int b); //passing int argument(parameters)
int main()
{
  printf("sum of two numbers=%d",sum(10,5));//passing int argument(parameters)
  return 0;

}
int sum(int a,int b)
{
  int c;
  c=a+b;
  return(c); //function passing return value
}
//Learn programming with Ankit Rami
```

```
sum of two numbers=15
--------------------------------
Process exited after 0.006445 seconds with return value 0
Press any key to continue . . .
```

```
Compilation results...
--------
- Errors: 0
- Warnings: 0
- Output Filename: D:\Programming-C AR\withargumentandWITHreturningvalue.exe
- Output Size: 128.486328125 KiB
- Compilation Time: 0.17s
```

Nested Functions in C

- ✓ Nested Functions are nothing but a function defined inside another function.
- ✓ **A function calling another function within its function definition is known as a nested function**.
- ✓ The scope of a nested function lies within the function in which it is declared and defined (enclosing function).
- ✓ Any local function, variable, constant, type, class, etc., that is in the same scope or any enclosing scope can be accessed by a nested function.
- ✓ This is the basic difference between nested functions and any other function, i.e., they have access to, and control over variables set out in their parent functions.
- ✓ Nested function definitions cannot access local variables of surrounding blocks. They can access only global variables. In C there are two nested scopes the local and the global. So nested function has some limited use.

```
main()
{
   clrscr();
   func1();
   getch();
}
void func1()
{
   for(i = 1; i<= 10; i++)
   {
      func2();
   }
}
void func2()
{
   printf("%d\n",i);
}
```

Simple Nested Functions in C

D:\Programming-C AR\nestedfunctioninc.cpp - [Executing] - Dev-C++ 5.11

```c
#include<stdio.h>
void sum();
void ar();
int main()
{
    sum();
}
void sum()
{
    int a=5;
    int b=10;
    int c=a+b;
    printf("Addition and is: %d ",c);
    ar();
}
void ar()
{
    printf("Learn programming with Ankit Rami");
}
//Learn programming with Ankit Rami
```

D:\Programming-C AR\nestedfunctioninc.exe

```
Addition and is: 15 Learn programming with Ankit Rami
--------------------------------
Process exited after 0.006435 seconds with return value 0
Press any key to continue . . .
```

```
Compilation results...
--------
- Errors: 0
- Warnings: 0
- Output Filename: D:\Programming-C AR\nestedfunctioninc.exe
- Output Size: 128.48828125 KiB
- Compilation Time: 0.16s
```

Simple Nested Functions in C

D:\Programming-C AR\nestedfunctioninc-2.cpp - [Executing] - Dev-C++ 5.11

```c
#include<stdio.h>
void sum();
int ar();
int a,i;
int main()
{
    sum();
}
void sum()
{
    for(a=1;a<=10;a++)
    {
        printf("%d \n",ar());
    }
}
int ar()
{
    return(a);
}
//Learn programming with Ankit Rami
```

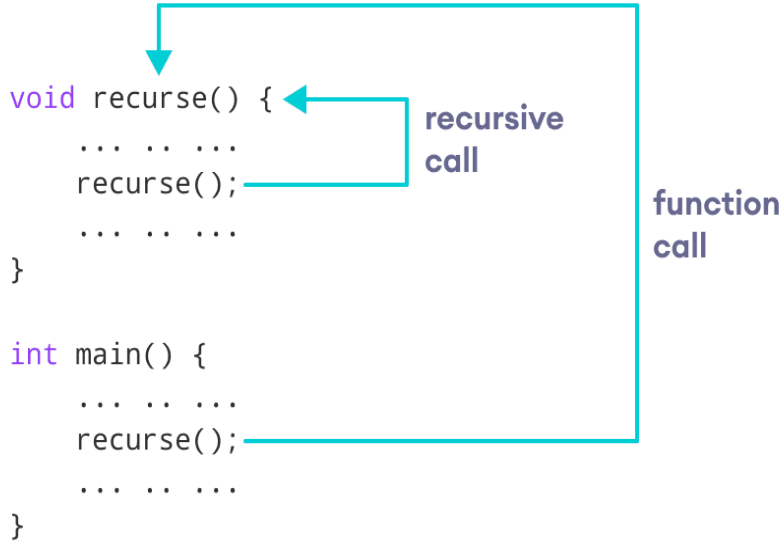
D:\Programming-C AR\nestedfunctioninc-2.exe

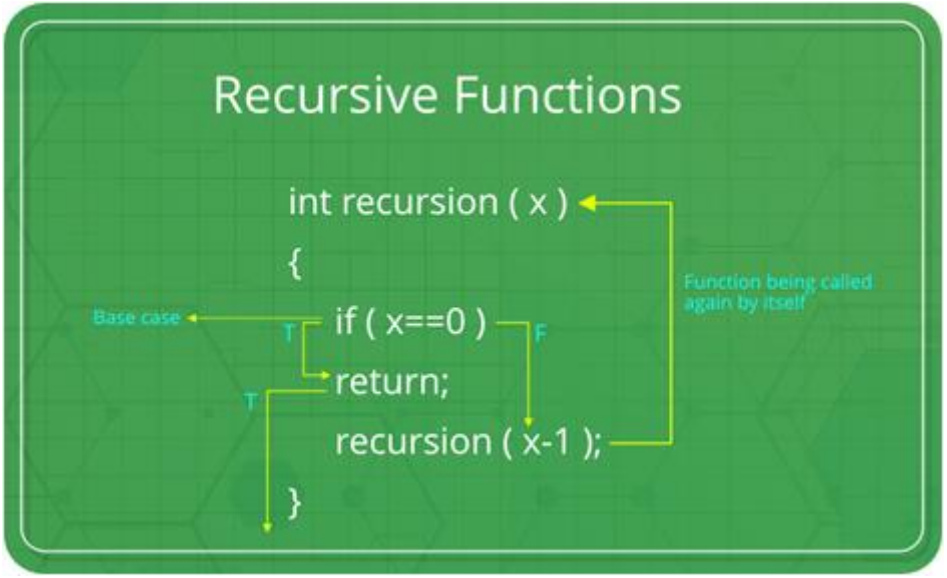
```
1
2
3
4
5
6
7
8
9
10

--------------------------------
Process exited after 0.01372 seconds with return value 0
Press any key to continue . . .
```

```
Compilation results...
--------
- Errors: 0
- Warnings: 0
- Output Filename: D:\Programming-C AR\nestedfunctioninc-2.exe
- Output Size: 128.525390625 KiB
- Compilation Time: 5.14s
```

Recursion in C

- ✓ A function that calls itself is known as a recursive function. And, this technique is known as recursion.
- ✓ Recursion cannot be applied to all the problem, but it is more useful for the tasks that can be defined in terms of similar subtasks. For Example, recursion may be applied to sorting, searching, and traversal problems.
- ✓ Recursive functions are very useful to solve many mathematical problems, such as calculating the factorial of a number, generating Fibonacci series, etc.





Simple Recursion Example in C

```c
#include <stdio.h>
int factorial(int i)
{
    if(i<=1)
    {
        return(1);
    }
    return(i*factorial(i-1));
}
int main()
{
    int i = 5;
    printf("Factorial of %d is %d\n", i, factorial(i));
    return 0;
}
```

```
Factorial of 5 is 120

--------------------------------
Process exited after 0.00551 seconds with return value 0
Press any key to continue . . .
```

📣 Types of Recursion in C

✓ Direct Recursion

- Direct recursion in C occurs when a function calls itself directly from inside. Such functions are also called direct recursive functions.
- Ex-

```
function_01()
{
    //some code
    function_01();
    //some code
}
```

✓ Indirect Recursion

- Indirect recursion in C occurs when a function calls another function and if this function calls the first function again. Such functions are also called indirect recursive functions.
- Ex-

```
function_01()
{
    //some code
    function_02();
}

function_02()
{
    //some code
   function_01();
}
```

Direct Recursion Example in C

D:\Programming-C AR\directrecursion.cpp - [Executing] - Dev-C++ 5.11

File Edit Search View Project Execute Tools AStyle Window Help

TDM-GCC 4.9.2 64-bit Release

(globals)

Project Classes Debug | [*] nestedfunctioninc.cpp | UDF1.c | [*] nestedfunctioninc-2.cpp | directrecursion.cpp

```c
#include<stdio.h>
int fibonacci(int i)
{
  if(i==0)
  {
    return 0;
  }
  if(i==1)
  {
    return 1;
  }
  return(fibonacci(i - 1) + fibonacci(i - 2));

}
int main()
{
  int i, n;
  printf("Enter a digit for fibonacci series: ");
  scanf("%d", & n);
  for (i = 0; i < n; i++) {
    printf(" %d ", fibonacci(i));
  }

  return 0;
}
```

Select D:\Programming-C AR\directrecursion.exe

```
Enter a digit for fibonacci series: 10
 0  1  1  2  3  5  8  13  21  34
--------------------------------
Process exited after 1.932 seconds with return value 0
Press any key to continue . . .
```

Compiler | Resources | Compile Log | Debug | Find Results | Close

Abort Compilation

Shorten compiler paths

```
Compilation results...
--------
- Errors: 0
- Warnings: 0
- Output Filename: D:\Programming-C AR\directrecursion.exe
- Output Size: 128.65234375 KiB
- Compilation Time: 0.13s
```

Line: 24 Col: 12 Sel: 0 Lines: 25 Length: 362 Insert Done parsing in 0.015 seconds

Type here to search | 21°C Smoke | ENG | 11:22 17-02-2024

Indirect Recursion Example in C

D:\Programming-C AR\IndirectRecursion.cpp - [Executing] - Dev-C++ 5.11

File Edit Search View Project Execute Tools AStyle Window Help

TDM-GCC 4.9.2 64-bit Release

(globals)

Project Classes Debug | [*] nestedfunctioninc.cpp | UDF1.c | [*] nestedfunctioninc-2.cpp | directrecursion.cpp | IndirectRecursion.cpp

```c
#include<stdio.h>
void odd();
void even();
int n=1;
void odd()
{
    if(n<=10)
    {
        printf("%d ", n+1);
        n++;
        even();
    }
    return;
}
void even()
{
    if(n<=10)
    {
        printf("%d ", n-1);
        n++;
        odd();
    }
    return;
}
int main()
{
    odd();
}
```

Select D:\Programming-C AR\IndirectRecursion.exe

```
2 1 4 3 6 5 8 7 10 9
--------------------------------
Process exited after 0.006974 seconds with return value 0
Press any key to continue . . .
```

Compiler | Resources | Compile Log | Debug | Find Results | Close

Abort Compilation

Shorten compiler paths

```
Compilation results...
--------
- Errors: 0
- Warnings: 0
- Output Filename: D:\Programming-C AR\IndirectRecursion.exe
- Output Size: 128.505859375 KiB
- Compilation Time: 0.16s
```

Line: 27 Col: 11 Sel: 0 Lines: 28 Length: 328 Insert Done parsing in 0 seconds

Type here to search | 21°C Smoke | ENG | 11:25 17-02-2024