Function

Prepared By: Tasmiah Tamzid Anannya, Lecturer, Cs Dept, AIUB

Example of a Function

```
#include <stdio.h>

void addNumbers();          // function prototype

int main()
{
    addNumbers();           // function call
    return 0;
}

void addNumbers()           // function definition
{
    int n1,n2;
    printf("Enters two numbers: ");
    scanf("%d %d",&n1,&n2);
    int result;
    result = n1+n2;
    printf("sum = %d",result);
}
```

Returning value from a function

```
#include <stdio.h>

int addNumbers();                    // function prototype

int main()
{
    int sum=addNumbers();            // function call
    printf(“%d”,sum);
    return 0;
}

int addNumbers()                     // function definition
{
    int n1,n2;
    printf("Enters two numbers: ");
    scanf("%d %d",&n1,&n2);
    int result;
    result = n1+n2;
    return result;                    //return statement
}
```

Receiving parameters

```
#include <stdio.h>

int addNumbers(int a, int b);          // function prototype

int main()
{   int n1,n2,sum;
    printf("Enters two numbers: ");
    scanf("%d %d",&n1,&n2);
    sum = addNumbers(n1, n2);          // function call
    printf("sum = %d",sum);
    return 0;
}

int addNumbers(int a,int b)            // function definition
{
    int result;
    result = a+b;
    return result;                     // return statement
}
```

Receiving parameters

```
#include <stdio.h>

int addNumbers(int a, int b);          // function prototype

int main()
{   int n1,n2,sum;
    printf("Enters two numbers: ");
    scanf("%d %d",&n1,&n2);
    sum = addNumbers(n1, n2);          // function call
    printf("sum = %d",sum);
    return 0;
}

int addNumbers(int a,int b)            // function definition
{
    int result;
    result = a+b;
    return result;                     // return statement
}
```

Receiving parameters

```
#include <stdio.h>

int addNumbers(int a, int b);          // function prototype

int main()
{   int n1,n2,sum;
    printf("Enters two numbers: ");
    scanf("%d %d",&n1,&n2);
    sum = addNumbers(n1, n2);          // function call
    printf("sum = %d",sum);
    return 0;
}

int addNumbers(int a,int b)            // function definition
{
    int result;
    result = a+b;
    return result;                     // return statement
}
```

Example

- The function prototype is not needed if the user-defined function is defined before the main() function.

Example

- The function prototype is not needed if the user-defined function is defined before the main() function.

```
#include <stdio.h>
int addNumbers(int a,int b)          // function definition
{
    int result;
    result = a+b;
    return result;                   // return statement
}

int main()
{   int n1,n2,sum;
    printf("Enters two numbers: ");
    scanf("%d %d",&n1,&n2);
    sum = addNumbers(n1, n2);        // function call
    printf("sum = %d",sum);
    return 0;
}
```

Some important points about function

- Every C program has a function called main() that is called by operating system when a user runs the program.

Some important points about function

- Every C program has a function called main() that is called by operating system when a user runs the program.

- 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.

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

Some important points about function

- 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)".***

Some important points about function

- 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)".***

- As a side note, in C++, empty list means function can only be called without any parameter. In C++, both void fun() and void fun(void) are same.

Some important points about function

- 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)".***

- As a side note, in C++, empty list means function can only be called without any parameter. In C++, both void fun() and void fun(void) are same.

- If in a C program, a function is called before its declaration then the C compiler automatically assumes the declaration of that function in the following way:
  int function name();
  And in that case if the return type of that function is different than **INT** ,compiler would show an error.

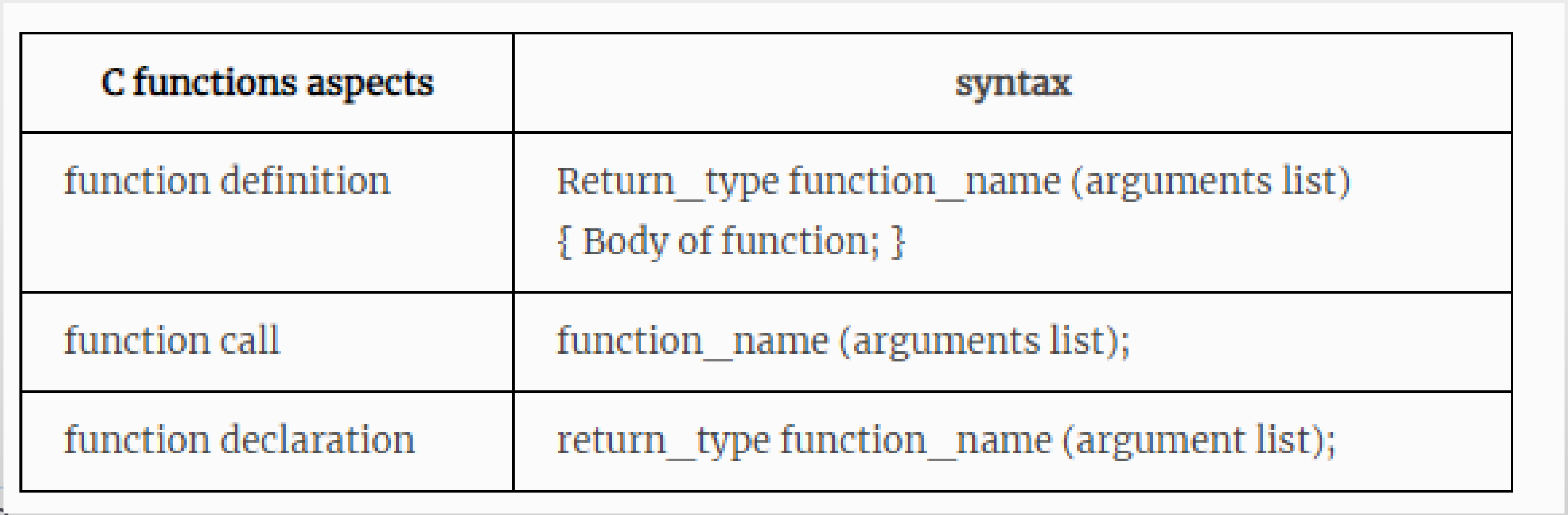
Example

```
#include <stdio.h>

int main()
{   int n1,n2;
    double sum;
    printf("Enters two numbers: ");
    scanf("%d %d",&n1,&n2);
    sum = addNumbers(n1, n2);        // function call
    printf("sum = %f",sum);
    return 0;
}

double addNumbers(int a,int b) {
    double result;
    result = a+b;
    return result;}
```

ERROR!!! Because forward declaration of the function Prototype is missing before main function, so the return type is automatically considered ***int***.

C FUNCTION DECLARATION, FUNCTION CALL AND FUNCTION DEFINITION

- There are 3 aspects in each C function. They are,
  - **Function declaration or prototype** – This informs compiler about the function name, function parameters and return value's data type.
  - **Function call** – This calls the actual function
  - **Function definition** – This contains all the statements to be executed.



C functions aspects	syntax
function definition	Return_type function_name (arguments list) { Body of function; }
function call	function_name (arguments list);
function declaration	return_type function_name (argument list);

Practice

- Write a program that contains a function named **square** which will take one float number as argument and will return the square value of that number to the **main** function.

HOW TO CALL C FUNCTIONS IN A PROGRAM?

- There are two ways that a C function can be called from a program-
  - Call by value
  - Call by reference

CALL BY VALUE:

- In call by value method, the value of the variable is passed to the function as parameter.
- The value of the actual parameter can not be modified by formal parameter.
- Different Memory is allocated for both actual and formal parameters. Because, value of actual parameter is copied to formal parameter.
- **Note**:
- **Actual parameter** – This is the argument which is used in function call.
- **Formal parameter** – This is the argument which is used in function definition

CALL BY VALUE

```
#include<stdio.h>
// function prototype, also called function declaration
void swap(int a, int b);
int main()
{
    int m = 22, n = 44;
    // calling swap function by value
    printf(" values before swap  m = %d \nand n = %d", m, n);
    swap(m, n);
}
void swap(int a, int b)
{
    int tmp;
    tmp = a;
    a = b;
    b = tmp;
    printf(" \nvalues after swap m = %d\n and n = %d", a, b);
}
```

HOW TO CALL C FUNCTIONS IN A PROGRAM?

CALL BY REFERENCE:

- In call by reference method, the address of the variable is passed to the function as parameter.
- The value of the actual parameter can be modified by formal parameter.
- Same memory is used for both actual and formal parameters since only address is used by both parameters.

CALL BY REFERENCE:

```
#include<stdio.h>
// function prototype, also called function declaration
void swap(int *a, int *b);
int main()
{
    int m = 22, n = 44;
    //  calling swap function by reference
    printf("values before swap m = %d \n and n = %d",m,n);
    swap(&m, &n);
}
void swap(int *a, int *b)
{
    int tmp;
    tmp = *a;
    *a = *b;
    *b = tmp;
    printf("\n values after swap a = %d \nand b = %d", *a, *b);
}
```

- All C functions can be called either with arguments or without arguments in a C program. These functions may or may not return values to the calling function. Now, we will see simple example C programs for each one of the below.
  - C function with arguments (parameters) and with return value.
  - C function with arguments (parameters) and without return value.
  - C function without arguments (parameters) and without return value.
  - C function without arguments (parameters) and with return value.