Unit IV – Part 02 (Function)

Lecture Slide

AS2023

In pursuit of preparing tomorrow's technologists

Objectives

By the end of this session, students will be able to:

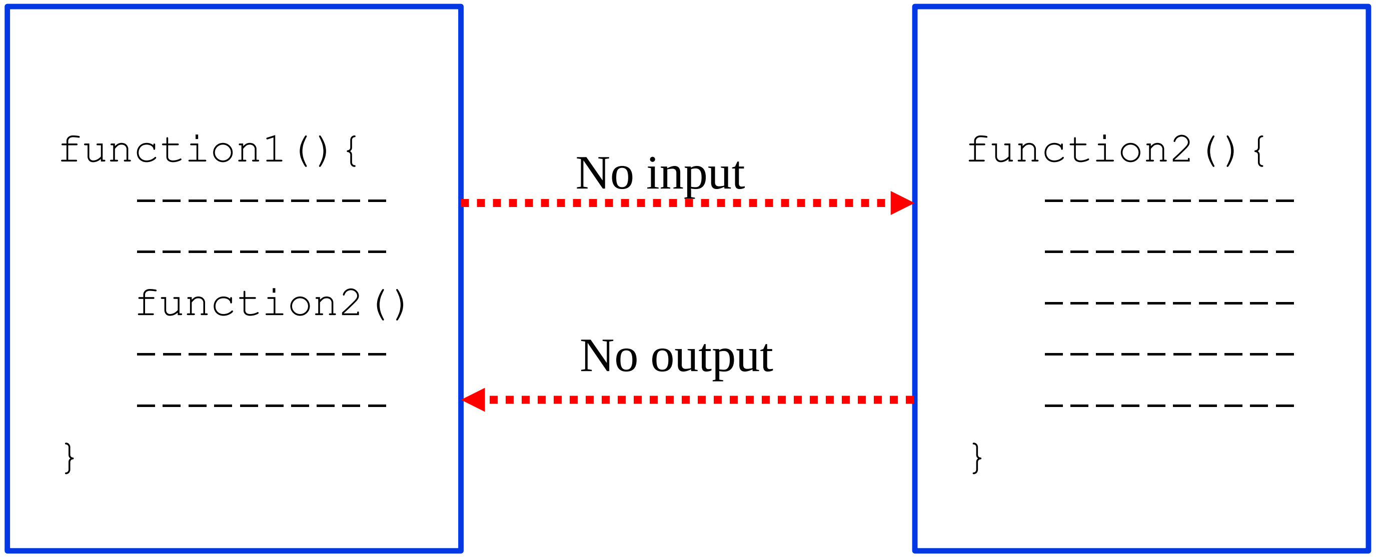
- Identify categories of function
- Explain each of the categories
- Choose appropriate categories
- Define scope, lifetime & visibility of an variable
- Explain various storage class

Categories of Function

- Based on the return values and parameter
- Broad categories
  - Functions with no arguments and no return values
  - Functions with arguments and no return values
  - Functions with arguments and one return value
  - Functions with no arguments but return a value
  - Functions that returns multiple values

Functions with no arguments and no return values

- No data transfer between the calling and called functions
- But only transfer of control exist



```
function1(){
    ----------
    ----------
    function2()
    ----------
    ----------
}
```

```
function2(){
    ----------
    ----------
    ----------
    ----------
    ----------
}
```

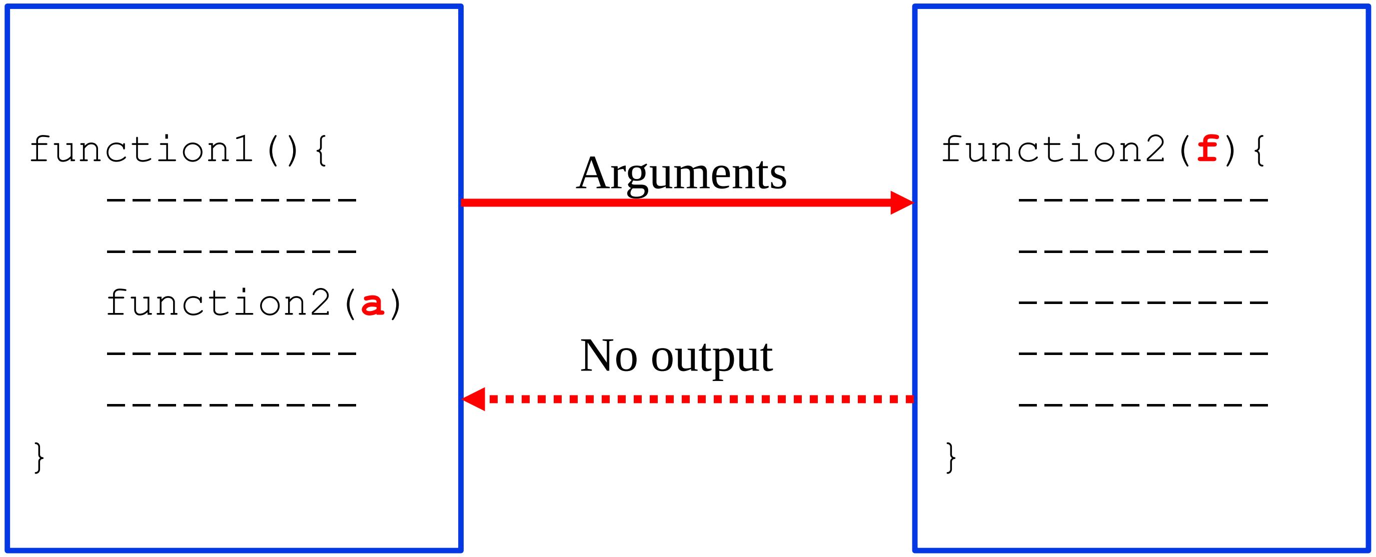
- **Note:** function that doesn't return any value cannot be used in an expression

Functions with no arguments and no return values

```
#include <stdio.h>
void println(void);
int main()
{
    println();
    printf("\n illustration of function in C\n");
    println();
    return 0;
}
void println(void){
    int i;
    for (i =0;i<40;i++){
        printf("_");
    }
}
```

Functions with arguments and no return values

- Data communication between two functions with argument but no return values



```
function1(){
    ----------
    ----------
    function2(a)
    ----------
    ----------
}
```

```
function2(f){
    ----------
    ----------
    ----------
    ----------
    ----------
}
```

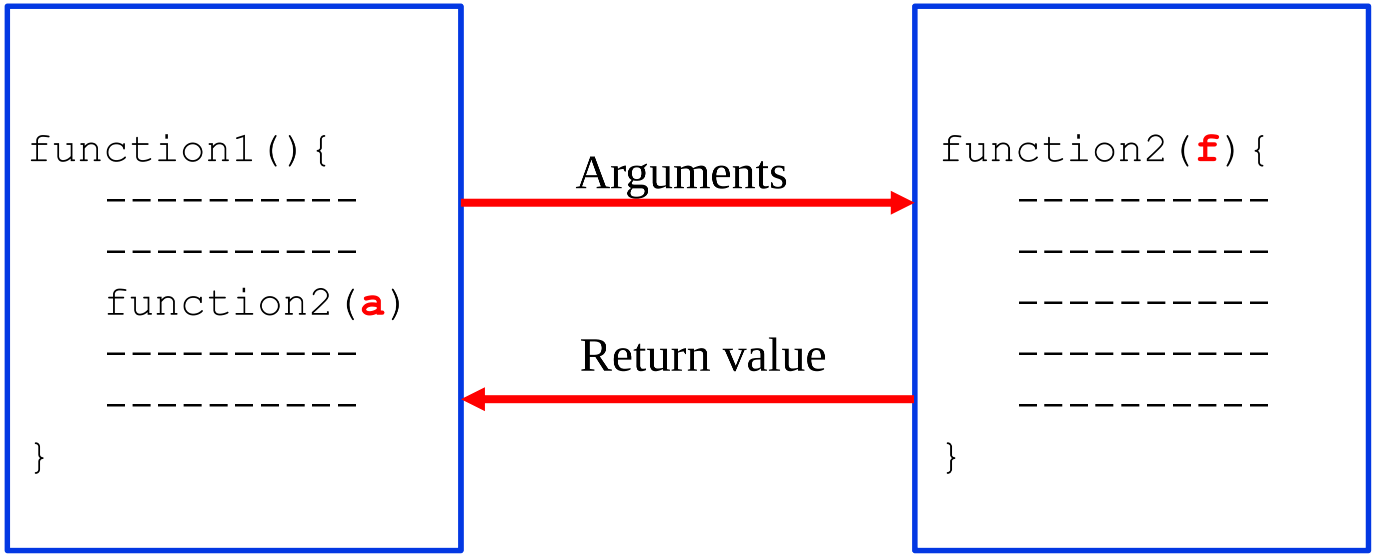
- Actual and formal parameter should match in number, order and type

Functions with arguments and no return values

```
#include <stdio.h>
void sum(int x, int y);
int main()
{
    int a=5, b=6, c;
    sum(a,b);
    return 0;
}
void sum(int x, int y){
    int s;
    s = x+y;
    printf("the sum is %d",s);
}
```

Functions with arguments and return values

- Data communication between two functions with argument and return values



```
function1(){
    ----------
    ----------
    function2(a)
    ----------
    ----------
}
```

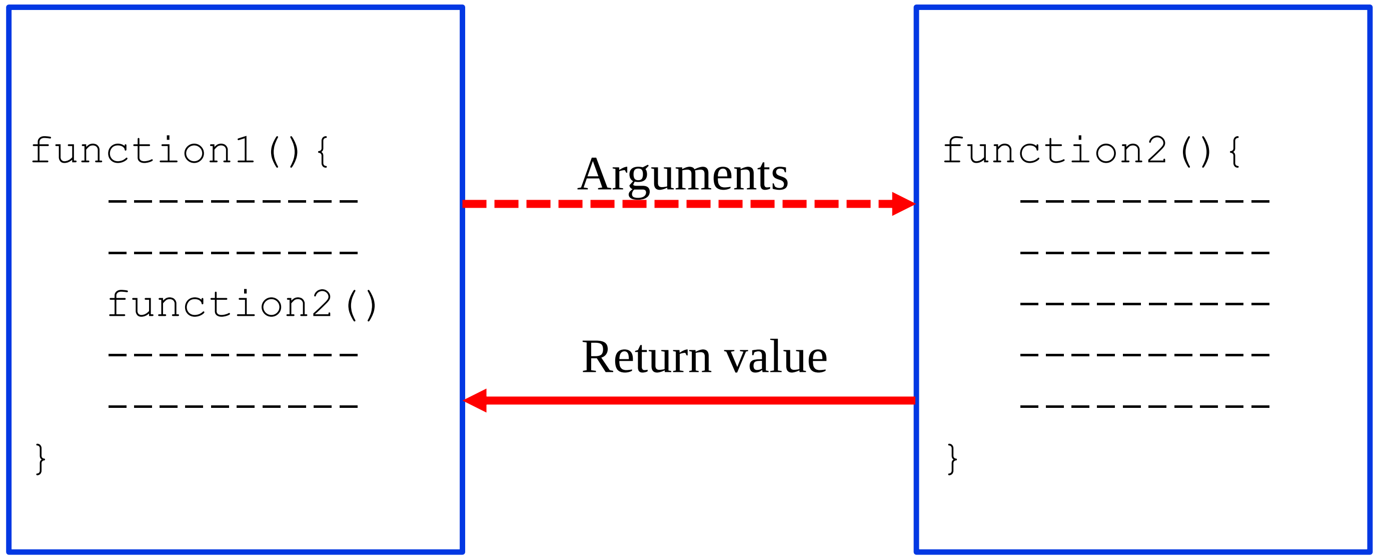
```
function2(f){
    ----------
    ----------
    ----------
    ----------
    ----------
}
```

Functions with arguments and return values

```
#include <stdio.h>
int sum(int x, int y);
int main()
{
    int a=5, b=6, c;
    c = sum(a,b);
    printf(“the sum is %d”,c);
    return 0;
}
int sum(int x, int y){
    int s;
    s = x+y;
    return s;
}
```

Functions with no arguments but returns a value

- Data communication between two functions with return values

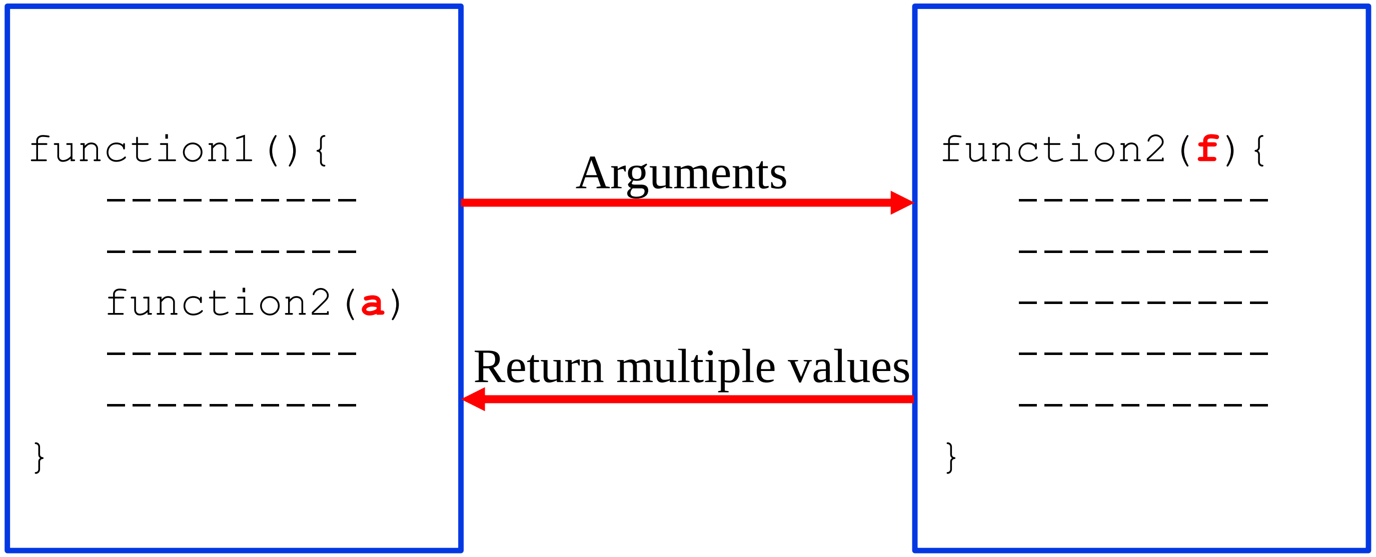


Functions with no arguments but returns a value

```
#include <stdio.h>
int sum(void);
int main()
{
    int c = sum();
    printf(“the sum is %d”,c);
    return 0;
}
int sum(void){
    int a=5, b=6, s;
    s = a+b;
    return s;
}
```

Functions that returns multiple values

- The arguments used to send out information are called *output parameters*



- Achieved by using address operator (&) and indirection operators (*)

Functions that returns multiple values

```
void mathoperation(int x, int y, int *s, int *d);
int main(){
    int x=20, y=10, s,d;
    mathoperation(x, y, &s, &d);
    printf("s=%d d=%d", s,d);
}
void mathoperation(int a, int b, int *s, int *d){
    *s = a+b;
    *d = a-b;
}
```

Scope, Visibility and Lifetime of a variable

- *Scope* determines the region for which the variable is accessible
- *Visibility* determines the accessibility of the variable from the memory
- *Longevity or lifetime* refers to the period during which the variable retains the given value during the execution of the program
- Different storage class:
  - Automatic
  - External
  - Static
  - Register

Home assignment

- Write short notes on the following storage class with the sample program for each
  - Automatic
  - External
  - Static
  - register

Call by Value & Call by Reference

- Functions can be called in two ways:
  1. **Call by Value**
  - The values of actual parameters are copied to formal parameters
  - Two copies of parameters stored in different memory location
  - Any changes made inside the function are not reflected in actual parameters

Call by Value & Call by Reference

2. Call by reference

- The address of actual parameter is passed to the function as formal parameters
- Both actual and formal parameters refer to the same location
- Any changes made inside the function are reflected in the actual parameters

Class Activity [10 Minutes]

1. Divide into groups (4 members)
2. Write the following functions without any argument and with no return type:
    1. Function to add three numbers
    2. Function to multiply three numbers
    3. Function to subtract two numbers
    4. Function to divide two numbers

3. Explain the concept to the class

Class Activity [5 Minutes]

1. Write the following functions with arguments and with no return type:
   1. Function to add three numbers
   2. Function to multiply three numbers
   3. Function to subtract two numbers
   4. Function to divide two numbers

3. Explain the concept to the class

Class Activity [5 Minutes]

1. Write the following functions with arguments and with return type:

   1. Function to add three numbers
   2. Function to multiply three numbers
   3. Function to subtract two numbers
   4. Function to divide two numbers

3. Explain the concept to the class

Class Activity [5 Minutes]

2. Write the following functions with no arguments and with return type

   1. Function to add three numbers
   2. Function to multiply three numbers
   3. Function to subtract two numbers
   4. Function to divide two numbers

3. Explain the concept to the class

Royal University of Bhutan

College of Science and Technology

Thank you