EDU

EAST DELTA UNIVERSITY

# Final Assignment

**Course title:** Structured Programming Laboratory

**Course Code:** CSE 113


**Submitted by,**

**Name:** Mohammad Fahim

**ID:** 242002112

**Section:** 4

**Department:** CSE

**Submitted to,**

**Name:** ANIKA BUSHRA

**Designation:** Lecturer, SoSET


**Date Of Submission:** 08-01-2024

# Answer to the question no 1

In C, a function can only return a single value technically. But we can return multiple values using pointer, structure or array.

Using pointer: We can use pointer to modify variable outside the function and effectively "return" multiple values.

Example:

```
#include <stdio.h>

void calculate(int a, int b, int *sum, int *product)
{
  *sum = a + b;
  *product = a*b;
}
  int main() {
    int num1 = 10, num2 = 5, sum, product;
    calculate(num1, num2, &sum, &product);
    printf("sum: %d\n", sum);
    printf("product: %d\n", product);
    return 0;
}
```

Output:

sum : 15

product : 50

Using structure: We can define a structure that holds multiple values and return them from the function.

Example:

```
#include <stdio.h>
typedef struct {
    int sum;
    int product;
} results;
results calculate (int a, int b) {
    results res;
    res.sum = a+b;
    res.product = a*b;
    return res;
}
int main() {
    int a=5, b=10;
    results result = calculate (a,b);
printf ("Sum : %d \n", result.sum);
```

```
    printf("Product: %d\n", result.product);
    return 0;
}
```

Output:

Sum: 15

Product: 50

## Answer to the question no 02

A function prototype is a declaration of a function that specifies its name, return type and parameters. It helps the compiler validate function calls.

### Purpose of Function Prototypes:

1. Enables calling functions before their definitions.
2. Ensures type checking for arguments and return values.
3. Promotes modularity by allowing function declarations in header files.
4. Improves code readability and maintainabil

## Define a Function prototype:

The general syntax of a function prototype:

return_type function_name (parameter list);

here,

- return_type is the type of the value function return.
- function_name is the name of the function.
- Parameter list is the data type of each parameter.

## Example:

```
#include <stdio.h>
int add (int, int);
int main (){
    int num1 = 5, num2 = 7;
    int result;
    result = add (num1, num2);
    printf("The sum of %d and %d is %d.\n", num1, num2,
                                    result);
    return 0;
}
  int add (int a, int b)
{
  return a+b;
}
```

Output:

Sum: 813

## Answer to the question no 03

The differences between passing an array to a function and passing a single-valued data item to a function are:

| Feature | Passing an array | Passing a single-value |
|---|---|---|
| Data Passed | A copy of the value | A pointer to the first element of the array |
| Parameter type | Primitive data type | Array or pointer |
| Access to Original data | Not possible directly | Direct access to the original data. |
| Function Behavior | Works on the copy of the variable | Works directly on the original data |
| Memory usage | Requires more memory for copying the value | Efficient; only the pointer is passed |

Example:

```
#include <stdio.h>
void modifyvalue (int x) {
    x = x * 2;
    printf("Inside modifyvalue, x = %d\n", x);
}
void modifyarray (int arr[], int size) {
    for(int i=0; i<size; i++) {
        arr[i] = arr[i] * 2;
    }
}
int main (){
    int value = 5;
    int arr[] = {1, 2, 3, 4, 5};
    int size = sizeof(arr)/sizeof(arr[0]);
    printf("Before modifyvalue, value = %d\n", value);
    modifyvalue (value);
    printf("After modifyvalue, value = %d\n", value);
    printf("\n Before modifyarray, arr = ");
```

```
    for(int i=0; i<size; i++) {
    printf("%d", arr[i]);
}
    printf("\n");
    modifyarray(arr, size);
    printf("After modifyarray, arr=");
    for(int i=0; i<size; i++) {
    printf("%d", arr[i]);
}
    printf("\n");
    return 0;
}
```

Output:

Before modify value, value= 10

Inside modifyvalue, n = 20

After modifyvalue, value= 10

Before modifyvalue, arr= 1, 2 3

After modifyvalue, arr = 2 4 6

## Answer to the question no 04

```
#include <stdio.h>

int find(char arr[], char target, int index) {
    if (arr[index] == '\0') {
        return 0;
    }
    if (arr[index] == target) {
        return 1;
    }
    return find(arr, target, index+1);
}

int main() {
    char str[] = "hello";
    char ch = 'e';
    if (find(str, ch, 0)) {
        printf("Character '%c' found in the array.\n", ch);
    } else {
        printf("Character '%c' not found in the array.\n", ch);
    }
    return 0;
}
```

## Answer to the question no 05

```
#include <stdio.h>
#include <string.h>

void copystring(char source[], char desti[]){
    int i=0;
    while(source[i] != '\0'){
        desti[i] = source[i];
        i++;
    }
    desti[i] = '\0';
}
void sortarray(int arr[], int size){
    for(int i=0; i<size-1; i++){
        for(int j=i+1; j<size; j++){
            if(arr[i] > arr[j]){
                int temp = arr[i];
                arr[i] = arr[j];
                arr[j] = temp;
            }
        }
    }
}
```

```
int main(){
    char source[] = "Hello, World!";
    char desti[50];
    copystring(source, desti);
    printf("Source string: %s\n", source);
    printf("copied string: %s\n", desti);
    int arr[] = {5, 2, 9, 1, 5, 6};
    int size = sizeof(arr) / sizeof(arr[0]);
    printf("Original Array: ");
    for(int i=0; i<size; i++){
        printf("%d", arr[i]);
    }
    printf("\n");
    sortarray(arr, size);
    printf("Sorted Array: ");
    for(int i=0; i<size; i++){
    printf("%d", arr[i]);
    }
    printf("\n");
    return 0;
}
```

## Answer to the question no 06

```
#include <stdio.h>

void add(int *a, int *b, int *result) {
    *result = *a + *b;
}

int main() {
    int num1 = 5, num2 = 10, sum;
    add(&num1, &num2, &sum);
    printf("Sum: %d\n", sum);

    return 0;
}
```

## Answer to the question no 07

```
#include <stdio.h>
#include <string.h>

struct date {
        int day, month, year;
    }
struct student {
        char name[50];
        struct date dob;
        int totalmarks;
      }
void sortstudents(struct student students[], int n) {
      for(int i=0; i<n-1; i++) {
         for (int j=i+1; j<n; j++){
       if(students[i].totalmarks < students[i].totalmarks) {
        students[i] = students[j];
        students[j] = temp;
      }
    }
  }
}
```

```
int main () {
    struct student students [10];
    for (int i=0; i<10; i++) {
    printf ("Enter name, DOB(dd mm yyyy), and total
        marks for student %d :", i+1);
    scanf ("%s %d %d %d %d", students[i]. name,
        & students[i]. dob-day & students[i] . dob. month,
        &students[i]. dob. year, & students [i] . totalmarks) ;
    }
sort students (students, 10);
    printf ("In Students Ranked by Marks : \n ");
    for (int i=0; i<10; i++) {
    printf("%s %.02d/%.02d/%.04d %d \n", students[i]. name,
        students [i]. dob. day, students [i] . dob. month,
        students [i]. dob. year, students [i] . totalmarks);
    }
    return 0;
}
```

## Answer to the question no 08

```
#include <stdio.h>

int sumarray(int *arr, int size){

        int sum=0;

    for(int i=0; i<size; i++){

        sum += *(arr+i);

    }

    return sum;

}

int main(){

        int arr[5] = {1, 2, 3, 4, 5};

        int result = sumarray(arr, 5);

    printf("Sum of arr elements: %d\n", result);

    return 0;

}
```