WEEK 1 ASSIGNMENT

6 Assignments on the use of pointers, arrays and basic functionality in the C language

Nalule Grace Nakabuye

13.06.2024
FABLAB RWANDA

INTRODUCTION

A report on the first week assignment.

Assignment 1

Objective

Using pointers to change the value of an integer `x` via one pointer and read the new value back via another pointer.

In the `main` function, I declare an integer `x` and two pointers `p1` and `p2`.

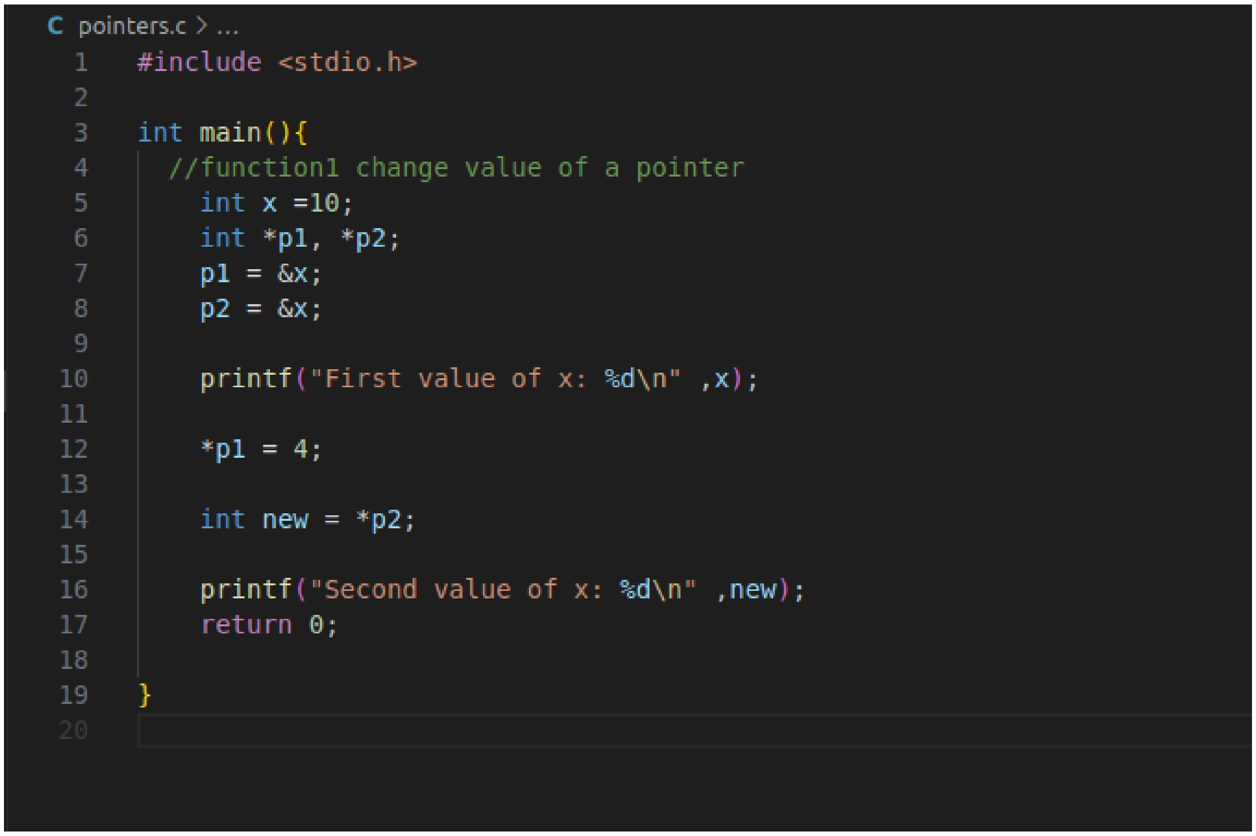
I store the address of `x` in both pointers `p1` and `p2`.

Print the initial value of `x`.

Use `p1` to change the value of `x`.

Declare a new integer “new” and assign it the value pointed to by `p2`.

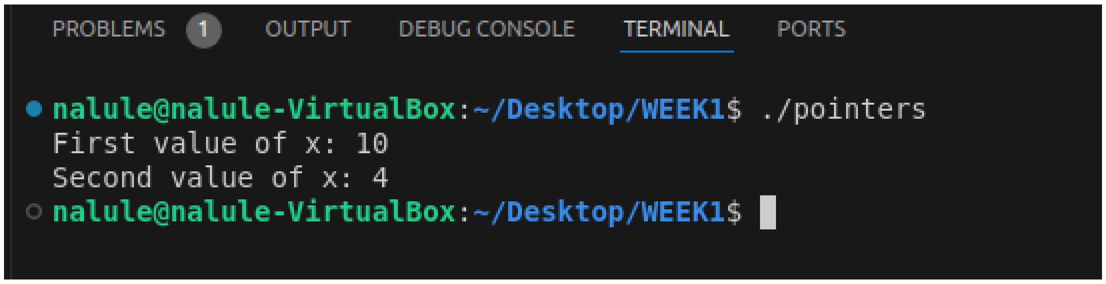
Print the value of `new`.



pointers.c > ...

```c
#include <stdio.h>

int main(){
  //function1 change value of a pointer
    int x =10;
    int *p1, *p2;
    p1 = &x;
    p2 = &x;

    printf("First value of x: %d\n" ,x);

    *p1 = 4;

    int new = *p2;

    printf("Second value of x: %d\n" ,new);
    return 0;

}
```

Output



PROBLEMS 1 OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
nalule@nalule-VirtualBox:~/Desktop/WEEK1$ ./pointers
First value of x: 10
Second value of x: 4
nalule@nalule-VirtualBox:~/Desktop/WEEK1$
```

Assignment 2

Objective

Using pointers to swap two values.

Steps

Declare a function swapValues that takes a pointer x and pointer y.

Declare an integer "now" a temporary integer where our pointer x points. We assign pointer x to pointer y and then pointer y to the variable "now".

//THIS IS THE FUNCTION DOING THE ACTUAL SWAPPING"

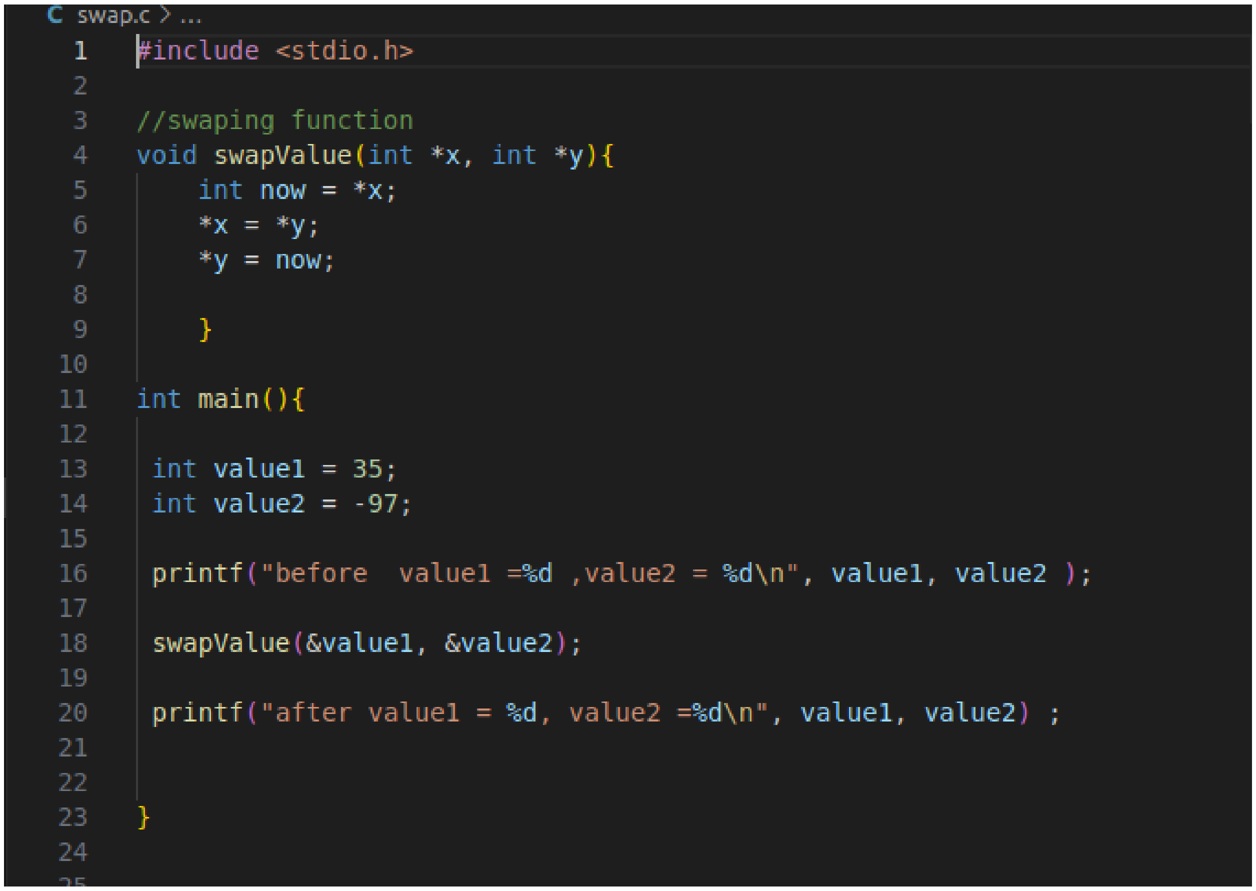
In the main, we parse two values value1=35 and value2=-97.

A print statement prints the before values of each respective value.

A call to the swap function that takes value1 and value2 as parameters.

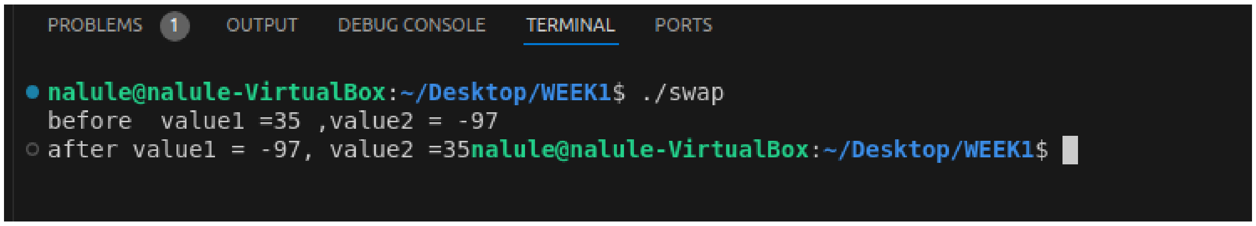
//This call initiates the swapping of values as per above function.

A print statement prints the after swap values respectively.

1. 

```c
#include <stdio.h>

//swaping function
void swapValue(int *x, int *y){
    int now = *x;
    *x = *y;
    *y = now;

    }

int main(){

 int value1 = 35;
 int value2 = -97;

 printf("before  value1 =%d ,value2 = %d\n", value1, value2 );

 swapValue(&value1, &value2);

 printf("after value1 = %d, value2 =%d\n", value1, value2) ;


}

```

Output

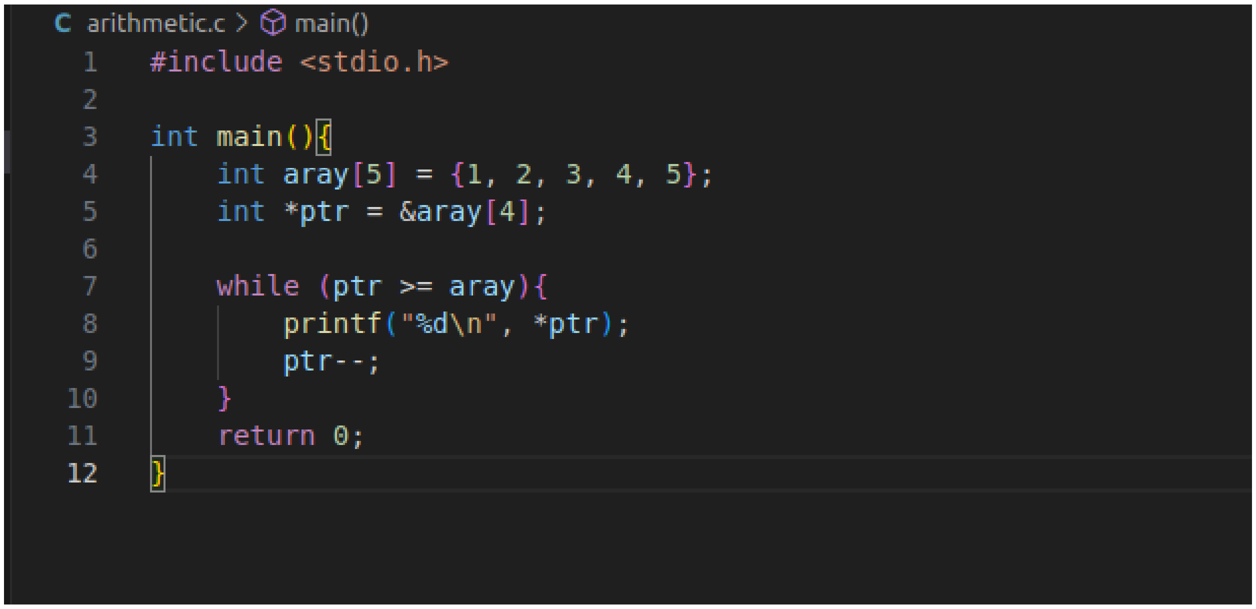


```
nalule@nalule-VirtualBox:~/Desktop/WEEK1$ ./swap
before  value1 =35 ,value2 = -97
after value1 = -97, value2 =35nalule@nalule-VirtualBox:~/Desktop/WEEK1$ 
```

Assignment 3

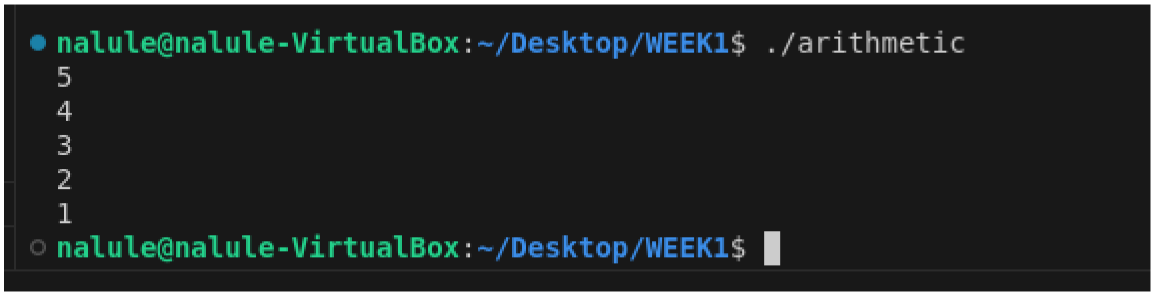
In the main function, an array 'aray' that takes 5 elements is declared. Followed by a pointer ptr that points to the last element in the array.

A while loop that takes the condition that ptr >= aray. And prints the value where pointer ptr is pointing. We then decrement ptr and the loop continues until our condition is met.



```
C arithmetic.c > main()
#include <stdio.h>

int main(){
    int aray[5] = {1, 2, 3, 4, 5};
    int *ptr = &aray[4];

    while (ptr >= aray){
        printf("%d\n", *ptr);
        ptr--;
    }
    return 0;
}
```

OUTPUT



```
nalule@nalule-VirtualBox:~/Desktop/WEEK1$ ./arithmetic
5
4
3
2
1
nalule@nalule-VirtualBox:~/Desktop/WEEK1$
```

Assignment 4

A function double arrayAdd that takes a pointer array and an int value.

A double variable sum is declared and equated to 0.0 (because it’s a double).

A for loop initiating i=0 and taking a condition that i< value then incrementing the value of i.

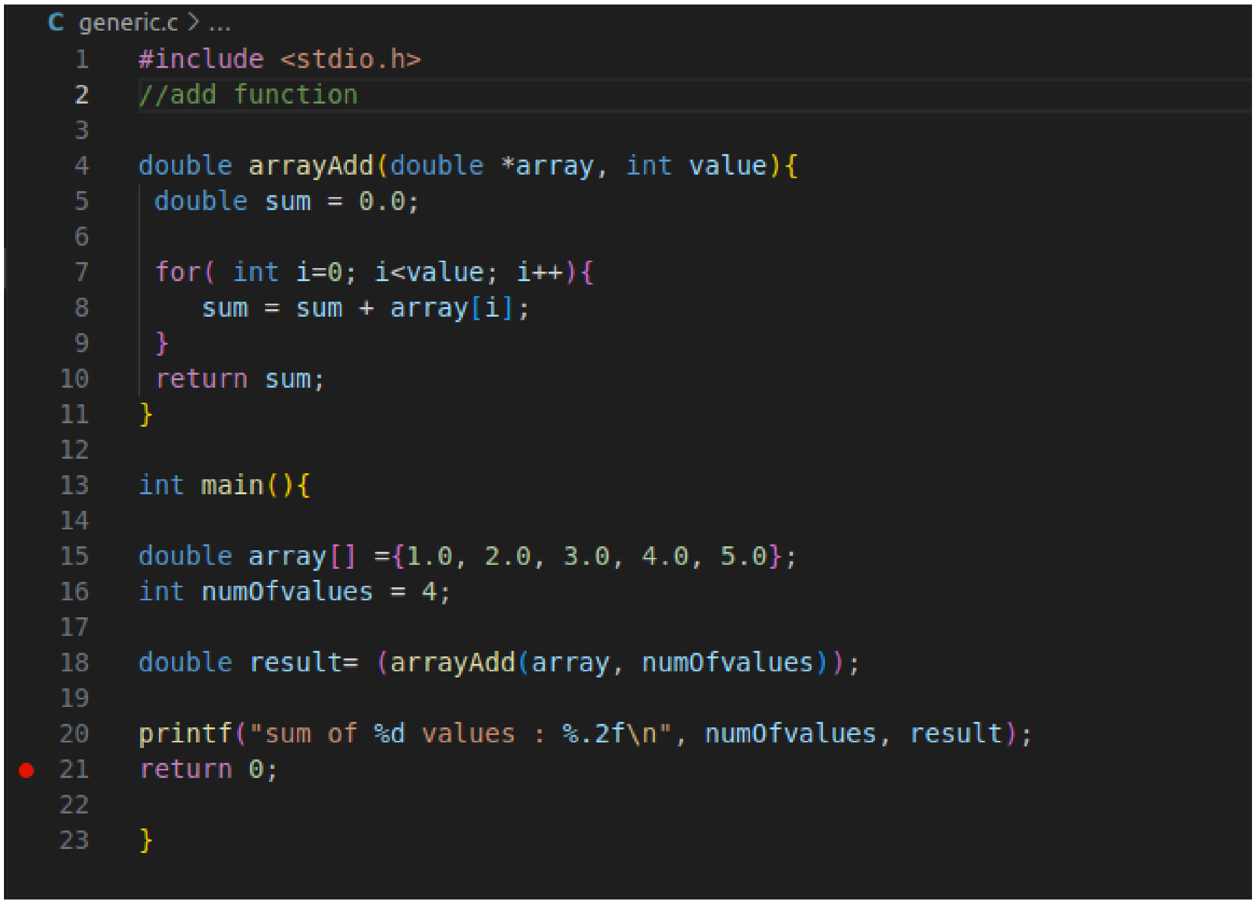
Sum equals sum plus the value of the element in the index "i" in our array. then we return sum;

//this is the function that performs summarizing.

In the main function, an array "array” that takes 5 elements is declared. And the number of values is specified as 4.

A double variable result is declared that calls our function "arrayAd”d and parses array and the specified number of values.

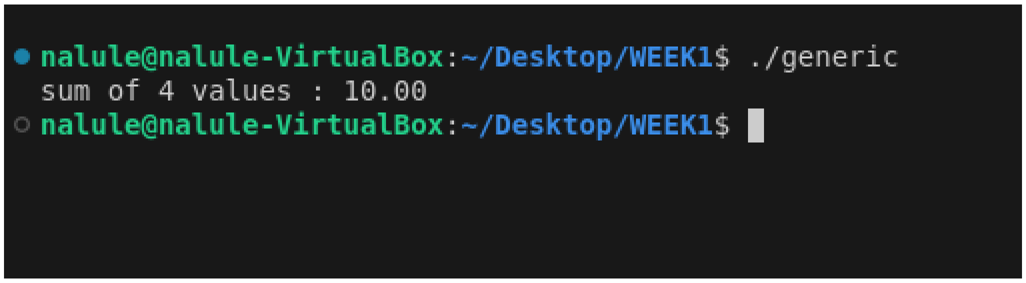
A print statement that prints the sum of specified values and then the result.



generic.c > ...

```c
#include <stdio.h>
//add function

double arrayAdd(double *array, int value){
 double sum = 0.0;

 for( int i=0; i<value; i++){
    sum = sum + array[i];
 }
 return sum;
}

int main(){

double array[] ={1.0, 2.0, 3.0, 4.0, 5.0};
int numOfvalues = 4;

double result= (arrayAdd(array, numOfvalues));

printf("sum of %d values : %.2f\n", numOfvalues, result);
return 0;

}
```

OUTPUT


```
● nalule@nalule-VirtualBox:~/Desktop/WEEK1$ ./generic
  sum of 4 values : 10.00
○ nalule@nalule-VirtualBox:~/Desktop/WEEK1$ 
```

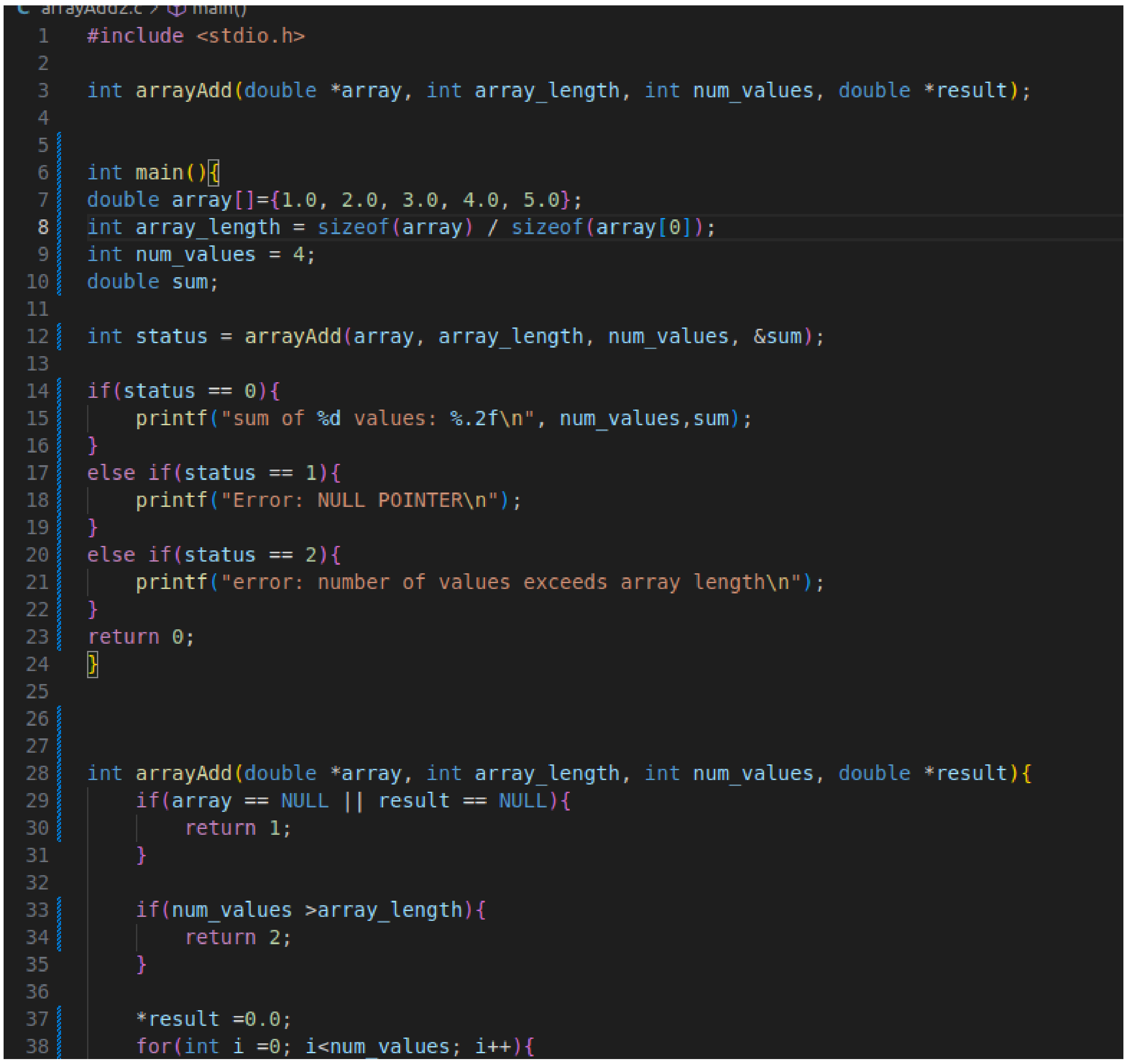
Assignment 5

Declare a function "arrayAdd" that parses a pointer "array" to the array of doubles, array_length(length of the array), num_values(number of values to sum) and "result" pointer to store the resulting sum.

The main function defines an array and calculates its length, num_values to specify how many values to sum. Sum to hold results and calls "arrayAdd" and checks the return status to handle errors appropriately.

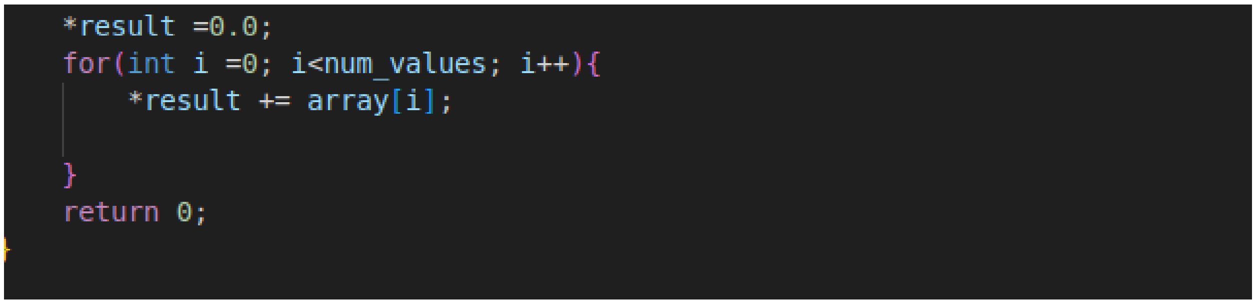
If 'array' or result is NULL return1, if num_values exceeds 'array_length' return 2, initialize *result to be 0.0.

Loop through the specified number of values in the array adding result to *result. Return 0.



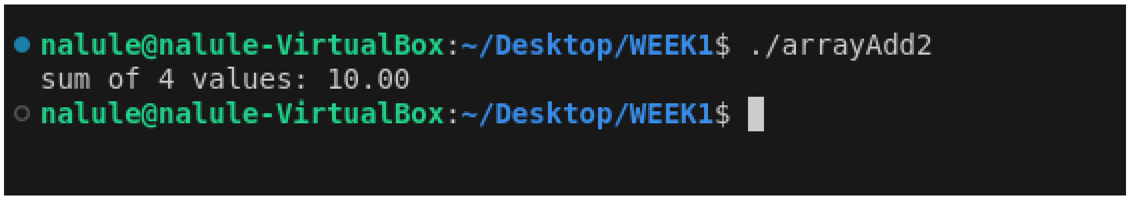
arrayAdd2.c > main()

```c
#include <stdio.h>

int arrayAdd(double *array, int array_length, int num_values, double *result);


int main(){
double array[]={1.0, 2.0, 3.0, 4.0, 5.0};
int array_length = sizeof(array) / sizeof(array[0]);
int num_values = 4;
double sum;

int status = arrayAdd(array, array_length, num_values, &sum);

if(status == 0){
    printf("sum of %d values: %.2f\n", num_values,sum);
}
else if(status == 1){
    printf("Error: NULL POINTER\n");
}
else if(status == 2){
    printf("error: number of values exceeds array length\n");
}
return 0;
}



int arrayAdd(double *array, int array_length, int num_values, double *result){
    if(array == NULL || result == NULL){
        return 1;
    }

    if(num_values >array_length){
        return 2;
    }

    *result =0.0;
    for(int i =0; i<num_values; i++){
```



```c
    *result =0.0;
    for(int i =0; i<num_values; i++){
        *result += array[i];

    }
    return 0;
}
```

OUTPUT



```
● nalule@nalule-VirtualBox:~/Desktop/WEEK1$ ./arrayAdd2
  sum of 4 values: 10.00
○ nalule@nalule-VirtualBox:~/Desktop/WEEK1$ 
```

REFERENCES

1. http://cslibrary.stanford.edu/106/