

JULY 1, 2021

PPS - ASSIGNMENT

WEEK 2

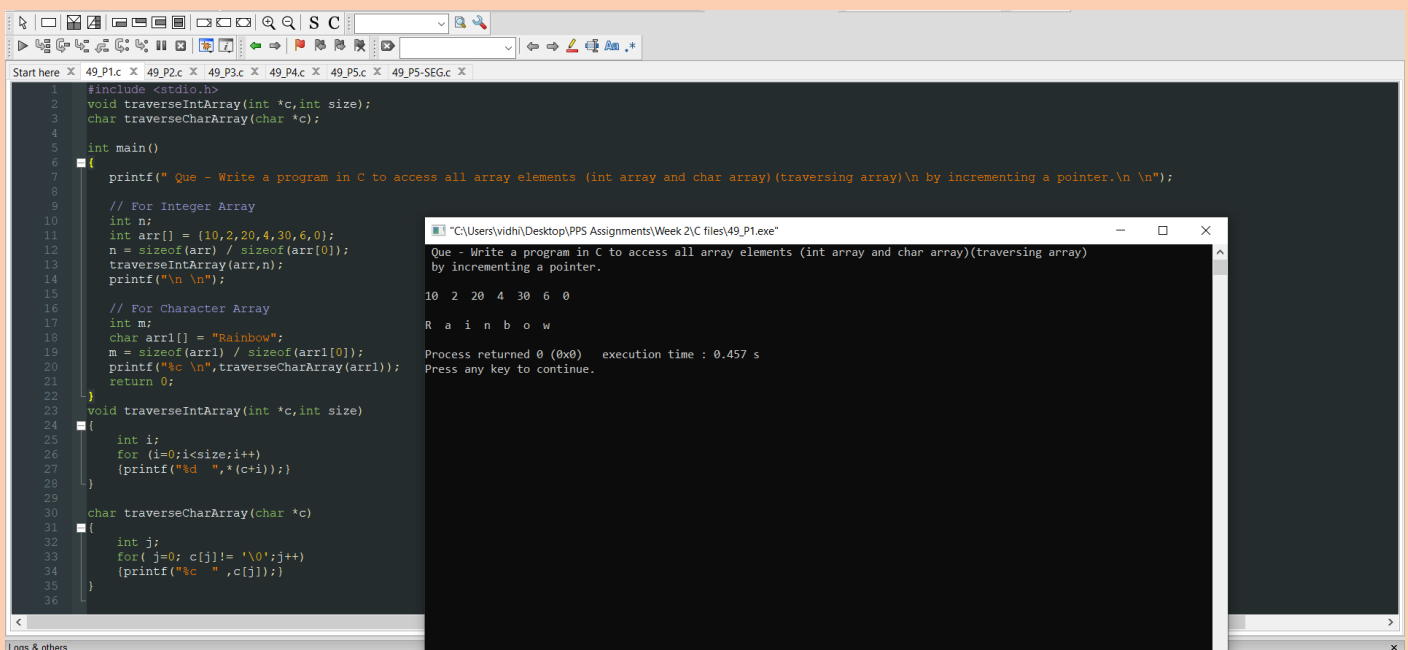
Prepared BY :
49_D1_CSE-Vidhi Bhatt

WRITE A PROGRAM IN C TO ACCESS ALL ARRAY ELEMENTS (INT ARRAY AND CHAR ARRAY) (TRAVERSING ARRAY) BY INCREMENTING A POINTER. EXPLAIN THIS CONCEPT IN DETAIL.

We here have first declared two functions, one for integer array and other for character array. Then inside main, we initialize both of our arrays. Next we take two variables m and n which stores the sizeof integer and character arrays respectively.

For accessing all elements of int array, the function `traverseIntArray` is called and the control is shifted to the the called function which is as defined after main. Here we try to access and print all elements using for loop. Variable `i` is initialized as 0 as we need to go till the last element. The size of array in our example is 7. As $0 < 7$ we enter the loop and the value at zeroth index is printed. Then `i` is incremented by 1. As $1 < 7$ we again enter the loop. This continues till $i < \text{size}$ becomes false. And thus the desired output is obtained.

For accessing all elements of char array, the function `traverseCharArray` is called and the control is shifted to function definition from main. Here we try to access and print all elements using for loop. Variable `j` is initialized as 0 as we need to go till the last element. The loop compares value of `j` with null character. (null character denotes end of char array). If `j != '\0'`, the char at zeroth index is printed and then `j` is incremented by 1. The loop goes on till null character is encountered and the required output is obtained.



The screenshot shows a C program in a code editor and its execution output in a separate window. The code defines two functions, `traverseIntArray` and `traverseCharArray`, which traverse arrays by incrementing a pointer. The `main` function initializes an integer array `arr1` and a character array `arr1` (containing "Rainbow"), and calls both traversal functions. The output window shows the execution results: the integer array elements (10, 2, 20, 4, 30, 6, 0) and the character array elements (R, a, i, n, b, o, w).

```
#include <stdio.h>
void traverseIntArray(int *c, int size);
char traverseCharArray(char *c);

int main()
{
    printf("Que - Write a program in C to access all array elements (int array and char array) (traversing array)\n by incrementing a pointer.\n\n");

    // For Integer Array
    int n;
    int arr1[] = {10, 2, 20, 4, 30, 6, 0};
    n = sizeof(arr1) / sizeof(arr1[0]);
    traverseIntArray(arr1, n);
    printf("\n\n");

    // For Character Array
    int m;
    char arr1[] = "Rainbow";
    m = sizeof(arr1) / sizeof(arr1[0]);
    printf("%c\n", traverseCharArray(arr1));
    return 0;
}

void traverseIntArray(int *c, int size)
{
    int i;
    for (i = 0; i < size; i++)
        (printf("%d ", *(c+i)));
}

char traverseCharArray(char *c)
{
    int j;
    for (j = 0; c[j] != '\0'; j++)
        (printf("%c ", c[j]));
}
```

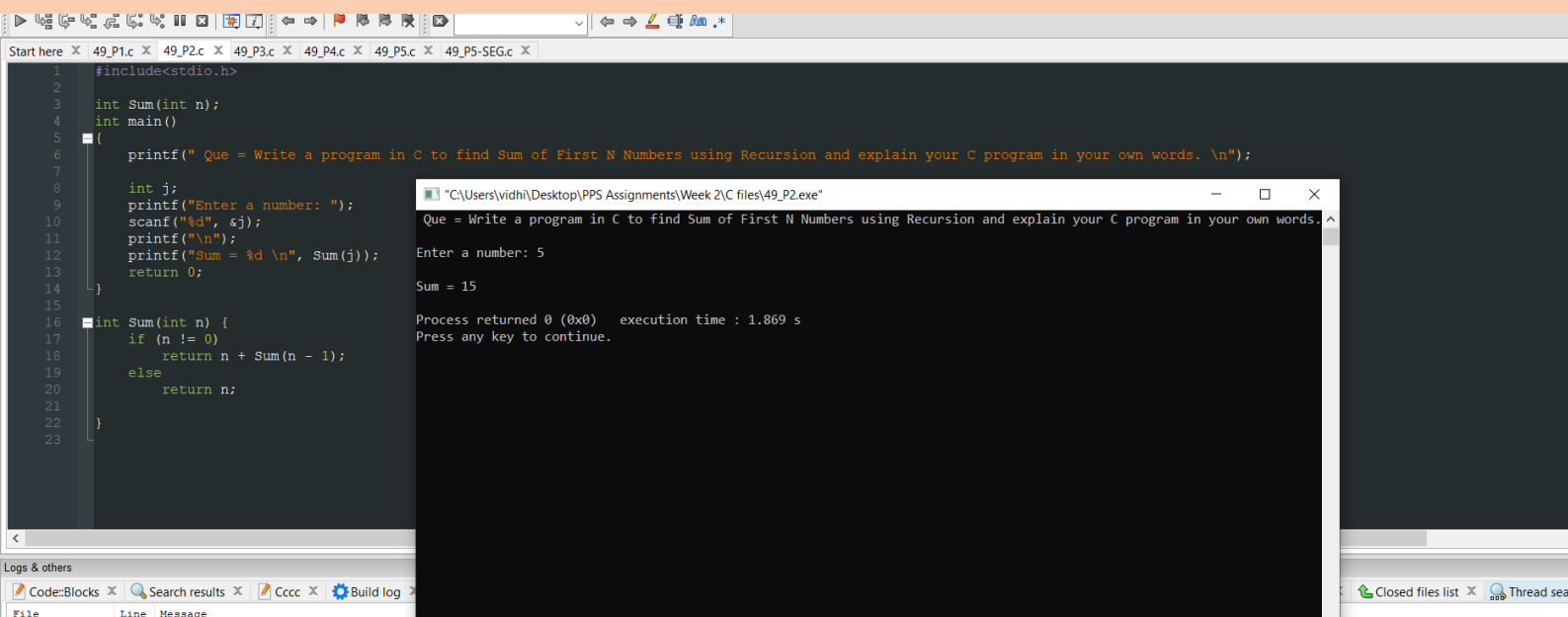
Que - Write a program in C to access all array elements (int array and char array) (traversing array) by incrementing a pointer.

10 2 20 4 30 6 0

R a i n b o w

Process returned 0 (0x0) execution time : 0.457 s
Press any key to continue.

WRITE A PROGRAM IN C TO FIND SUM OF FIRST N NUMBERS USING RECURSION AND EXPLAIN YOUR C PROGRAM IN YOUR OWN WORDS.



```
1 #include<stdio.h>
2
3 int Sum(int n);
4 int main()
5 {
6     printf(" Que = Write a program in C to find Sum of First N Numbers using Recursion and explain your C program in your own words. \n");
7
8     int j;
9     printf("Enter a number: ");
10    scanf("%d", &j);
11    printf("\n");
12    printf("Sum = %d \n", Sum(j));
13    return 0;
14 }
15
16 int Sum(int n) {
17     if (n != 0)
18         return n + Sum(n - 1);
19     else
20         return n;
21 }
22
23
```

Output Window: "C:\Users\vidhi\Desktop\PPS Assignments\Week 2\C files\49_P2.exe"

```
Que = Write a program in C to find Sum of First N Numbers using Recursion and explain your C program in your own words.
Enter a number: 5
Sum = 15
Process returned 0 (0x0)   execution time : 1.869 s
Press any key to continue.
```

FIRST, WE HAVE DECLARED A FUNCTION NAMED SUM WITH DATA TYPE INT. THEN INSIDE THE MAIN WE HAVE TAKEN A VARIABLE NAMED J WHOSE DATATYPE IS INT. NEXT WE ASK THE USER TO ENTER A NUMBER, THAT NUMBER IS STORED IN VARIABLE J. WE NEED TO CALCULATE THE SUM OF N NUMBERS WHICH BRINGS US ON OUR NEXT STATEMENT WHERE THE CONTROL IS SHIFTED TO THE SUM FUNCTION. LET'S ASSUME THAT NO. ENTERED BY USER IS 5. AS PER FUNCTION DEFINITION IT IS FIRST CHECKED IF OUR NO!=0 OR NOT. IF ITS TRUE THEN THE SUM IS RETURNED ELSE WE RETURN N. IN OUR EXAMPLE AS 5!=0, THE STATEMENT RETURNS 5+SUM(4). AGAIN NOW FUNCTION SUM(4) IS CALLED. THIS GOES ON TILL N=0. THE FINAL ANS IS THEN SENT BACK TO PRINT STATEMENT IN MAIN AND THE DESIRED RESULT IS OBTAINED.

QUE - 3

WRITE A PROGRAM IN C TO FIND WHETHER A NUMBER IS PRIME OR COMPOSITE USING RECURSION AND EXPLAIN YOUR C PROGRAM IN YOUR OWN WORDS.

We first declare a function named `checkPrime` with datatype `int`. In the main section we take 2 variables `a` and `b`. We ask the user to enter a number. This no. is stored in variable `a`. Prime nos. are those nos. which only have two factors 1 and the number itself. Let us understand the working of this program with the help of an example. Let the number that user entered be 7. Therefore the statement becomes `b=checkPrime(7,3)`. The control is shifted to the `checkPrime` function defined below. We always get 0 when a number is divided by it's factor. In our eg we first check if `3==1`, as this is false we move to the next if statement. Again here the condition `7%3==0` becomes false and we move to next statement. The function `checkPrime` is called again with the parameters `(7,2)`. This goes on till one of the if condition becomes true. The function `checkPrime` returns 1 and 0 for first and second if condition respectively. This value is sent to `b`. Then we use if else statements and print is Prime Number if `b=1` and is Composite Number if `b=0`.

```

1 #include<stdio.h>
2
3 int checkPrime();
4 int main()
5 {
6     printf(" Que = Write a program in C to find whether a Number is Prime OR Composite using Recursion \n \n");
7     int a,b;
8     printf(" Enter a positive number: ");
9     scanf(" %d", &a);
10    b=checkPrime(a,a/2);
11    if (b==1)
12        printf(" %d is a Prime Number \n",a);
13    else
14        printf(" %d is a Composite Number \n",a);
15    return 0;
16 }
17
18 int checkPrime(int a , int c)
19 {
20     if(c==1)
21         return 1;
22     if(a%c==0)
23         return 0;
24     return checkPrime(a,c-1);
25 }
26

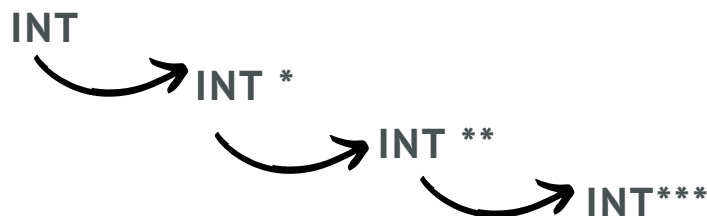
```

Process returned 0 (0x0) execution time : 1.667 s
Press any key to continue.

QUE - 4

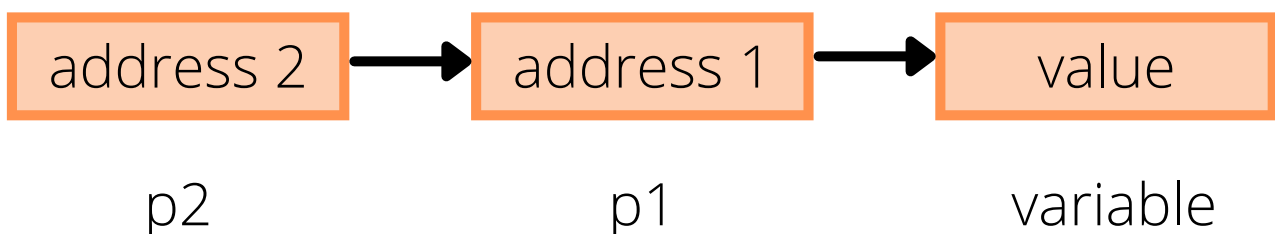
EXPLAIN THE CONCEPT OF POINTER TO POINTER WITH AN EXAMPLE.

IN C , IT IS POSSIBLE TO MAKE A POINTER POINT AT ANOTHER POINTER THUS CREATING A CHAIN OF POINTERS. AS A POINTER POINTS TO ANOTHER POINTER HERE, THIS CONCEPT IS ALSO KNOWN AS POINTER TO POINTER CONCEPT.



WHEN YOU USE A POINTER, THERE ARE TWO POSSIBILITIES. ONE IS THAT YOU GET A VALUE AT THAT LOCATION AND THE ANOTHER IS YOU GET AN ADDRESS OF SOME OTHER LOCATION.

LET US CLEAR OUR CONCEPT OF POINTER TO POINTER USING AN EXAMPLE.



QUE - 4 CONTINUE...

```

1 #include<stdio.h>
2
3 int main()
4 {
5     printf(" This is a short example to explain the concept of pointer to pointer. \n\n");
6     int n = 10;
7
8     int *p;
9     p=&n;
10    printf(" Address 1 = %d \n" ,p);
11    printf(" Value 1 = %d \n" ,*p);
12
13    int **p1;
14    p1=&p;
15    printf(" Address 1 = %d \n" ,p1);
16    printf(" Value 2 = %d \n" , **p1);
17
18    int ***p2;
19    p2=&p1;
20    printf(" Address 1 = %d \n" ,p2);
21    printf(" Value 3 = %d \n" , ***p2);
22
23    return 0;
24 }
25

```

```

This is a short example to explain the concept of pointer to pointer.

Address 1 = 6422036
Value 1 = 10
Address 1 = 6422024
Value 2 = 10
Address 1 = 6422016
Value 3 = 10

Process returned 0 (0x0)   execution time : 0.438 s
Press any key to continue.

```

In this example we have initialized a variable named `n` with datatype `int`. The value of `n` is 10. A pointer `*p` is used to point the variable `n` ie. the `p` stores the address of `n`. Similarly we have two more pointers `**p1` and `***p2` which point at pointers `*p` and `**p1` respectively.

Let us try to understand what exactly is happening here. Say for example we call `***p2`, at `***p2` we get the address of another location. We go to this pointed location(`**p1`) and we find out that there is another address which is being pointed at(`*p`). Now we go to this new address and find out that the pointer `*p` has a value of variable `n` which is 10. And hence the value 10 is returned.

In simple words you try to access the value at `***p`, when you reach there it tells you to go to `**p`, when you reach `**p` it tells you to go to `*p`, and finally when you reach `*p` it gives you the value at that address.

It is to be noted that the address of each of these pointers are different, but the value which they return is the same which can be clearly understood from the program.

QUE - 5

EXPLAIN MACRO IN DETAIL. WRITE A C PROGRAM TO FIND THE LARGEST ELEMENT IN AN ARRAY USING RECURSION AND MACRO. EXPLAIN THE PROGRAM EXECUTION IN YOUR OWN WORDS.

What are MACROS ?

MACROS in C are relatively very small functions in term of execution. They are generic. In other words, they are simple functions without any datatype. Generally MACROS are represented in capital letters.

Syntax of MACRO:

The syntax of MACRO is composed of mainly two parts. MACRO name and definition. The syntax is `#define function_name(Arguments) logic`.

Let us understand more about MACRO using a simple example. MACRO enables us to execute a function before the actual execution of program starts. In the below example we define a macro named as INCREMENT with argument x and logic `++x`. This basically helps us to increment the value of our integer b by 1 and in case of char array it changes the value base address.

The screenshot shows a C program in a code editor and its execution output in a terminal window. The code defines a macro `INCREMENT(x) ++x` and uses it to increment a character array `a` and an integer `b`. The output shows the string `Rainbow` shifted to `ainbow` and the integer `50` incremented to `51`.

```

1 #include<stdio.h>
2 #define INCREMENT(x) ++x
3
4 int main()
5 {
6     //small example to understand MACROS
7     char *a = "Rainbow";
8     int b = 50;
9
10    printf(" %s \n",a);
11    printf(" %s \n",INCREMENT(a));
12    printf(" %d \n",INCREMENT(b));
13
14    return 0;
15 }
16

```

Output of the program:

```

Rainbow
ainbow
51
Process returned 0 (0x0)   execution time : 0.429 s
Press any key to continue.

```

Build logs at the bottom show the compilation process:

```

File      Line  Message
=====
-- Build file: "no target" in "no project" (compiler: gcc)
-- Build finished: 0 error(s), 0 warning(s) (0 minute(s))

```

QUE - 6 CONTINUE...

The screenshot shows a C program in a code editor with the following code:

```

1 #include<stdio.h>
2 #define Max(x,y) ((x)>(y) ? (x) : (y))
3
4 int findMax(int *, int);
5 int main()
6 {
7     printf("Que = Write a C Program to find the largest element in an Array using Recursion and Macro.\n\n");
8
9     int S[] = {20, 1, 2, 30};
10    int size = sizeof(S)/sizeof(S[0]);
11    int ans = findMax(S, size);
12
13    printf("Maximum value is : %d\n", ans);
14    return 0;
15 }
16
17 int findMax(int *p, int size)
18 {
19     return (size==1)? *p: Max(*p, findMax(p+1, size-1));
20 }
21
22

```

The execution output shows the program running and displaying the following text:

```

Que = Write a C Program to find the largest element in an Array using Recursion and Macro.
Maximum value is : 30
Process returned 0 (0x0)   execution time : 0.411 s
Press any key to continue.

```

In this example we are going to find the largest element in an array using the concept of macro. Here we define a macro named MAX with two arguments x and y. As per the logic, if the condition is true the part before colon is executed else the part after. Next we define a function named findMax with datatype int. Then in the main we initialize an array S and then calculate the size of the array in variable size. In the next statement we call the function findMax in order to find the largest element.

The control is now shifted to the function findMax defined below. The value at base address and size of array are passed as arguments here. If the condition size==1 is true then *p is returned to ans else the macro MAX is executed. This goes on and on till the condition size==1 becomes true. Here we have assumed that the value at base address is max by default and then we compare it with other elements. If this is not true we move onto next element and repeat the above procedure. Finally the ans is printed using the printf statement.

#Advantages of MACRO:

- The length of program is reduced.
- Execution of program becomes faster as a lot of time is saved which compiler takes to invoke and call functions.

#Disadvantages of MACRO:

- They replace the code, they are not function calls.
- Program becomes lengthy if MACRO is called several times.



THANK
YOU