Tutorial Link https://codequotient.com/tutorials/Functions and Arrays/59fddb32e63d6b7fd5debfda

TUTORIAL

Functions and Arrays

Chapter

1. Functions and Arrays

Topics

While dealing with arrays, the array name itself corresponds to the base address of the array itself, where the first element is stored. We can pass the array variables to the functions just like other simple variables. Array variables are always passed by reference, because array is collection of element and it is not a good idea to make the copies of array in all functions. Now, if an array is an argument of a function, then at the time of calling of function, its base address will be passed to the function. The function can access the whole array with the base address. And also if a function returns an array then it will return the base address of the array. Be careful when returning an array from functions, as the array must remain in memory after the end of function execution, otherwise the calling program is not able to use the array. There are two possible ways to declare function with array in parameter list.

- We can either have an array in parameter.

```
int sum (int arr[]);
```

- Or, we can have a pointer in the parameter list, to hold base address of array.

```
int sum (int* ptr);
```

Following programs will use and explain this concept: -

```c
#include<stdio.h>
int sum(int a[]);               //declaration of the function with the parameter as an array
int main()  {
        int a[] = {1,2,3,4,5};
        int s;
        s = sum(a); //sum function returns an integer value and it is stored in int variable s
        printf("Sum of the array elements is %d ",s);
}

int sum(int a[])
{        //receiving array as a formal argument
        int i,total=0;
        for(i=0;i<5;i++) {
                total += a[i];
        }
        return total;           //returning an integer variable total
}

```

The output of the above program is

```
Sum of the array elements is 15
```

Also we don't return an array from functions, rather we return a pointer holding the base address of the array to be returned. But we must, make sure that the array exists after the function ends. For example,

```
int* sum (int x[]) {
        //statements
```

```
        return x ;
}
```

The following program will return an array from a function:

Return an array from a function

```c
#include <stdio.h>
int* fillValues( )
{
        static int  r[10];
                /* declared static so after function termination must remain in memory */
        int i;
        for ( i = 0; i < 10; ++i)
        {
                r[i] = i*i;
                printf( "r[%d] = %d\n", i, r[i]);
        }
        return r;       // returning the base address of array
}
int main ()
{
        int *p;         /* a pointer to an int */
        int i;
        p = fillValues();       // call function and hold the base address returned in pointer
        for ( i = 0; i < 10; i++ ) {
                printf( "*(p + %d) : %d\n", i, *(p + i));
        }
}
```

The output of above program is: -

```
r[0] = 0
r[1] = 1
r[2] = 4
r[3] = 9
r[4] = 16
r[5] = 25
```

```
r[6] = 36
r[7] = 49
r[8] = 64
r[9] = 81
*(p + 0) : 0
*(p + 1) : 1
*(p + 2) : 4
*(p + 3) : 9
*(p + 4) : 16
*(p + 5) : 25
*(p + 6) : 36
*(p + 7) : 49
*(p + 8) : 64
*(p + 9) : 81
```

Video Solution

<iframe width="560" height="315" src="https://www.youtube.com/embed/aFtyNvHy3PM" title="YouTube video player" frameborder="0" allow="accelerometer; autoplay; clipboard-write; encrypted-media; gyroscope; picture-in-picture" allowfullscreen></iframe>