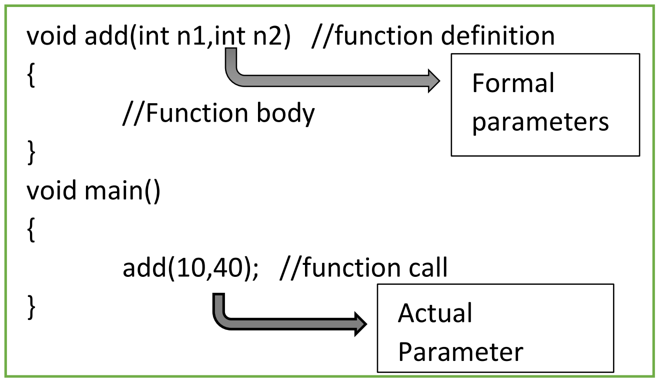
Principles of Programming (IMSC0206/IMCA0206)

Unit 3

Topics : actual parameter and formal parameter, Calling function by value and call by reference, Pointer, Pass array to function

Difference between actual parameter and formal parameter

Actual parameters are parameters as they appear in function calls. Formal parameters are parameters as they appear in function declarations.



```
void add(int n1,int n2)   //function definition
{
        //Function body
}
void main()
{
        add(10,40);   //function call
}
```

Here n1 and n2 are formal parameter while 10 and 40 are actual parameter.

Calling function by value and call by reference

There are two methods to pass the data into the function in C language

1. *call by value*
2. *call by reference*.

What is Call by Value?

Call by value method copies the value of an argument into the formal parameter of that function.

we cannot modify the value of the actual parameter by the formal parameter.

In this parameter passing method, values of actual parameters are copied to function's formal parameters, and the parameters are stored in different memory locations. So any changes made inside functions are not reflected in actual parameters of the caller.

Example:

```
#include<stdio.h>

void change(int num) {

   printf("Before adding value inside function num=%d \n",num);

   num=num+10;

   printf("After adding value inside function num=%d \n", num);

}

void main() {

   int x=20;

   printf("Before function call x=%d \n", x);

   change(x);//passing value in function

   printf("After function call x=%d \n", x);

}
```

Output

```
Before function call x=20

Before adding value inside function num=20

After adding value inside function num=30

After function call x=20
```

What is Call by reference?

In call by reference, the address of the variable is passed into the function call as the actual parameter.

The value of the actual parameters can be modified by changing the formal parameters since the address of the actual parameters is passed.

In call by reference, the memory allocation is similar for both formal parameters and actual parameters. All the operations in the function are performed on the value stored at the address of the actual parameters. Modified value gets stored at the same address.

Example:

```
void swap (int *a, int *b)  //function definition
{
   int temp;
   temp = *a;
   *a=*b;
   *b=temp;
   printf("After swapping values in function a = %d, b = %d\n",*a,*b); // Formal parameters, a =
20, b = 10
}

void main()
{
   int a = 70;
   int b = 20;
   printf("Before swapping the values in main a = %d, b = %d\n",a,b);    swap(&a,&b);
   printf("After swapping values in main a = %d, b = %d\n",a,b);
   getch();
}
```

No.	Call by value	Call by reference
1	A copy of the value is passed into the function.	An address of value is passed into the function.
2	Changes made inside the function is limited to the function only.	Changes made inside the function validate outside of the function also.
3	Actual and formal arguments are created with the different memory location.	Actual and formal arguments are created with the same memory location.

Pointers

What is pointer?

The pointer is a variable which stores the address of another variable. This variable can be of type int, char, array, function, or any other pointer. The size of the pointer depends on the architecture. However, in 32-bit architecture the size of a pointer is 2 byte.

type *var-name;

Example:

int no = 100;

int *p = &no; // Variable p of type pointer is pointing to the address of the variable n of type integer.

How to declare pointer:

We can declare pointer as following:

int *a; //pointer to int

char *c; //pointer to char

Address of (&) operator

```
#include<stdio.h>
void main()
{
        int a;
        clrscr();
        a=50;
        printf("value of number is %d, address of number is %u",a,&a);
        getch();
}
```

Example:

```
void main()
{
        int number=50;
        int *p;
        p=&number;                  //stores the address of number variable
        printf("Address of p variable is %x \n",p);  // p contains the address of the number
        therefore printing p gives the address of number.
        printf("Value of p variable is %d \n",*p);
        /* As we know that * is used to dereference a pointer therefore if we print *p, we will get
        the value stored at the address contained by p.    */
```

}

Advantages of Pointer

1) Pointer reduces the code and improves the performance.

2) We can return multiple values from a function using the pointer.

3) It makes us able to access any memory location in the computer's memory.

Passing Array to function

Array can also be passed to a function as an argument.

The array name is the address of first element of that array. For example if array name is arr then arr is equivalent to the &arr[0].

Example:

```
void fun( int *no1, int size)
{
        int x;
           for(x=0; x<size; x++)
         {
              printf("Value of arr[%d] is: %d \n", x, *no1);
              /*increment pointer for next element fetch*/
              no1++;
          }
}

void main()
{
   int arr[] = {1, 2, 3, 4, 5, 6, 7};
        clrscr();
   fun(arr, 6);
getch();
}
```

References:

https://www.javatpoint.com/c-pointers

https://www.geeksforgeeks.org/how-arrays-are-passed-to-functions-in-cc/

https://beginnersbook.com/2014/01/c-passing-array-to-function-example/