Content 30

Global, Local and Static Variable.

#### Global Variables

Global variables are defined outside a function, usually in the main. Global variables hold their values, and we can access them inside any of the functions defined for the program. Global variables are initialized by the system automatically when we define them.

*If the local and global variables have the same name, the local variable will take preference.*

#include<stdio.h>

int func1(){

    return 27;

}

int main()

{

    printf("The number in Func1() is : %d", func1());

    return 0;

}

**Output:**

The number in Func1() is : 27

But here I am now storing value as Valve,

1. //here i am storing my value in val
2. int func1(int b){
3. return b \* 10;
4. }
5. int main()
6. {   int b=34;
7. int val= func1(b);
8. int \*ptr= &val;
9. printf("The number in Func1() is : %d", val);
10. return 0;
11. }

**Output:**

The number in Func1() is : 340

int func1(int b){

this variable ab line 2 and

int val= func1(b);

this variable at line 7 both are different.

**Code1 for giving Both variables address:**( to show that both are different)

int func1(int b)

{

    printf("\nThe adress of b(at func1 body) here is: %d", &b);

    return b \* 10;

}

int main()

{

    int b = 34;

    printf("\nThe adress of b(at main body) here is: %d", &b);

    int val = func1(b);

    int \*ptr = &val;

    printf("\nThe number in Func1() is : %d", val);

    return 0;

}

**Output:**

The adress of b(at main body) here is: 6422216

The adress of b(at func1 body) here is: 6422192

The number in Func1() is : 340

*Here both the addresses are different. This means both the variables are different. That’s why whenever we call for any value then actual parameter doest go but its copy goes to formal parameters.*

Here I am Writing Program for the Explaining **Global Variable**;

I am declaring b as Global variable(as it is out of main body and func body

1. //Progam for Explaining Global Variable
2. #include <stdio.h>
3. int b=96;   //here I had declared b as global variable
4. int func(int b)
5. {
6. return 10 \* b;
7. }
8. int main(int argc, char const \*argv[])
9. {
10. int b = 45; //this b is local parameter
11. int val = func(b);
12. int \*ptr = &val;
13. printf("The Value of b is: %d\n",val);
14. return 0;
15. }

**Output:**

The Value of b is: 450

But still I am not getting the value by Global Variable because there is a formal parameter b in func.

Now I am going to change the line no 4 and 6 in the above program:

So now our program is like;

1. //Progam for Explaining Global Variable
2. #include <stdio.h>
3. int b = 96; //here I had declared b as global variable
4. int func(int b1)
5. {
6. printf("The Value of b as Global Parameter is: %d\n", b); //here will print the global parameter
7. return 10 \* b1;
8. }
9. int main(int argc, char const \*argv[])
10. {
11. int b = 45; //local variable of main.
12. int val = func(b);
13. int \*ptr = &val;
14. //func(b);
15. printf("The Value of b is: %d\n", val);
16. return 0;
17. }

**Output;**

The Value of b as Global Parameter is: 96

The Value of b is: 450

Now we get the value of our Global Variable, becoz we had changes formal parameter of func as b1 and there no Local Variable also. All here is Global Variable will take presidence.

#### Local Variables

Local variables are declared inside a function or a block of code; they cannot be accessed outside the function. The local variables can be used only by statements that are inside that function or block of code. The system does not initialize local variables, we must initialize it ourself. The scope of these variables will be within the function only.

Explaination about **Local Variable**;

int b = 45; //local variable of main.

Here at line 11 of above code, b is an Local variable of main body so it had its Identity in main body or function only not out of main body. All I mean is you cant print Local variable out of its own area or boby.

#### Formal Parameters

Formal parameters have precedence over global variables, and they are treated as local variables within a function. The term **formal parameters are used** to refer to the **parameters** in the definition of the method. In contrast, actual**parameters** are the variable or expression that appears in the function or method call in the calling environment.

Now let us move to our main topic, i.e., static variables. Static is a keyword in the C language. We can use it with variables and functions.

#### What is a static variable?

A static variable is known to retain the value even after they exit the scope. Static variables retain their value and are not initialized again in the new scope. The static variable until the end of the program is kept in the memory, whereas a normal variable is destroyed when a function is over. They can be defined inside or outside the function. Static variables are local to the block. The default value of static variables is zero. The keyword **static** is used to declare a static variable.

#### Syntax:

static Datatype Variable\_name = Variable\_value;

**datatype** − The data type of variable like char, int, float, etc.

**variable\_name** − This is the name of the variable.

**Variable\_value** − Value to initialize the variable. By default, it is zero.

// For Explaining Static ariables

#include <stdio.h>

int func(int b1)

{

    static int myvar = 4;

    printf("The value of b is: %d\n", myvar);

    myvar++;

    return 0;   //return = b1 +myvar;

}

int main()

{

    int a ;

    func(a);

    func(a);

    func(a);

    // int \*ptr = &a;

    return 0;

}

**Output:**

The value of b is: 4

The value of b is: 5

The value of b is: 6

Here I had entered func(a); 3 times in main body so I got 1st output line as initial value which I initialized in program’s func body with static and last 2 output lines with incremented values.

“How much times you entered func(a); in main body you will get incremented values that much times”.