Local Variable:

Variables declare in the function are called Local variables

abc.c

#include<stdio.h>

int main()

{

int x,y;

x = 10;

y = 20;

lmn(x,y);

}

void lmn(int q, int p){

int r;

r = q+p;

printf("%d",r);

}

| 700 | 10 | 800 | 20 | 900 | 10 | 925 | 20 | 950 | 30 |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |

When we define any variable then in the Ram there is a allocation table maintained

Example

| Program | function | Variable name | Base address | Number of bytes |
| --- | --- | --- | --- | --- |
| abc | main | x | 700 | 4 |
| abc | main | y | 800 | 4 |
| abc | lmn | p | 900 | 4 |
| abc | lmn | q | 925 | 4 |
| abc | lmn | r | 950 | 4 |

This table maintained with variable names address and size

Once the function ends local variable die

Or Memory is released (means entry removed from allocation table)

So, after

void lmn(int q, int p){

int r;

r = q+p;

printf("%d",r);

}

This line data will not be deleted just not reserved and can be allocated to anyone.

| Program | function | Variable name | Base address | Number of bytes |
| --- | --- | --- | --- | --- |
| abc | main | x | 700 | 4 |
| abc | main | y | 800 | 4 |
| ~~abc~~ | ~~lmn~~ | ~~p~~ | ~~900~~ | ~~4~~ |
| ~~abc~~ | ~~lmn~~ | ~~q~~ | ~~925~~ | ~~4~~ |
| ~~abc~~ | ~~lmn~~ | ~~r~~ | ~~950~~ | ~~4~~ |

If we call lmn function again then again memory will allocate and once function end again release. Once main end then

| Program | function | Variable name | Base address | Number of bytes |
| --- | --- | --- | --- | --- |
| ~~abc~~ | ~~main~~ | ~~x~~ | ~~700~~ | ~~4~~ |
| ~~abc~~ | ~~main~~ | ~~y~~ | ~~800~~ | ~~4~~ |

These allocated memories will also released.

Global Variable:

The variables that are defined outside the functions are global variables.

Scope: Can be used anywhere But after declaration. If suppose you write the function then you define any variable then you can use that variable after the declaration but can not use it in a function written before the declaration.

Memory Allocation: as soon as the program runs memory will be allocated,

However in the local variable, when the control comes to the function then only memory for the local variable will be allocated.

How to add comments in code

Single line comment: //

Multi-line comment : /\* … body … \*/

Comments will be ignored by the compiler.

Example:

/\*

This example demonstrates how a global variable is declared

and how it is available for all the functions within a program

\*/

#include<stdio.h>

//The following line is the declaration of sam

void sam(); // return type of sam is void and has zero parameters.

int x;

int main()

{

x = 10;

printf("%d\n", x);

sam();

printf("%d\n", x);

return 0;

}

void sam()

{

x = 350;

}

As we know once the function ends local variable dies.

When the program ends then Global variable dies.

What if we give same name to local and global variable

#include<stdio.h>

void sam();

int x;

int main()

{

x = 10;

printf("%d\n",x);

sam();

printf("%d\n",x);

return 0;

}

void sam()

{

int x;

x = 203;

printf("%d\n",x);

}

/\*\*

\* output

10

203

10

\*/

It means when we define the same name for global and local variable

Then priority will be given to the local variable.

However, that local assignment will not affect the global values.

Real-life example:

Suppose you have a pet named Tommy and in your city, there is another dog with the same name Tommy. when mom says go and give this bread to Tommy then you will first give it to your pet, if that does not exist then you will go for colony dog.

The same happens here priority will be given to local over global.

But what if Mom says give this bread to colony Tommy?

I mean how to call a global variable inside the function where the local variable is already present with the same name?

See an example:

#include<stdio.h>

void sam();

int x;

int main()

{

x = 10;

printf("%d\n",x);

sam();

printf("%d\n",x);

return 0;

}

void sam()

{

int x; // local x

x = 203; // local x

printf("%d\n",x); // local x

{

extern int x; // memory won't be allocated for x because of keyword extern

printf("%d\n", x); // global x

x=5050; // global x

}

printf("%d\n", x); // local x

}

/\*\*

\* output

10

203

10

203

5050

\*/

To use Global variable in function we have to use in in block {}

That will be separate independent block in which we can use with keyword Extern

And due to that keywords memory will not be allocated to that variable.