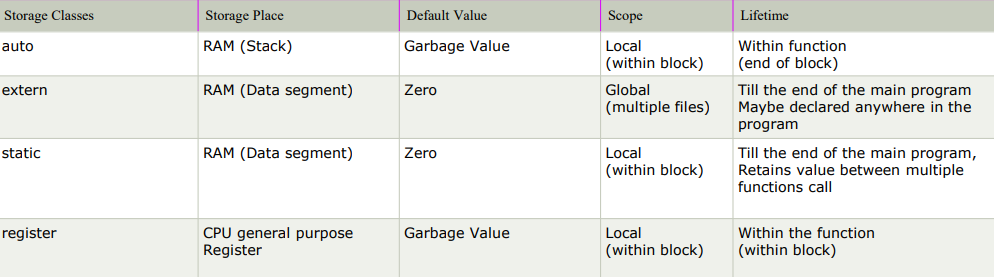
**Storage Classes in C**

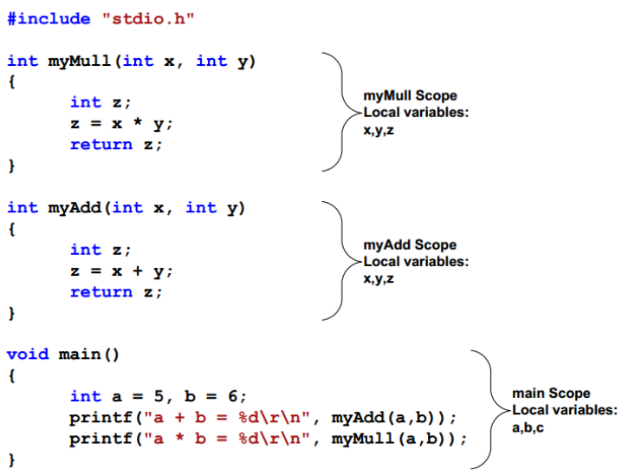
Storage classes in C are used to determine the lifetime, visibility, memory location, and initial value of a variable. There are four types of storage classes in C

* Automatic
* External
* Static
* Register



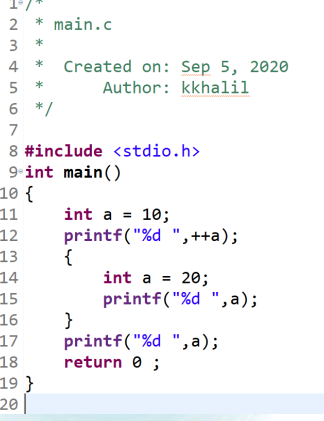
## Automatic

* Automatic variables are allocated memory automatically at runtime
* This is the default storage class for all the variables declared inside a function or a block.
* The visibility of the automatic variables is limited to the block in which they are defined
* The scope of the automatic variables is limited to the block in which they are defined
* Every local variable is automatic in C by default.
* Also it can be accessed outside their scope as well using the concept of pointers given here by pointing to the very exact memory location where the variables resides.
* The automatic variables are initialized to garbage by default.

Above program have three functions each with different scopes; each scope holds different local variables.

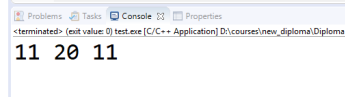
The variable a, b and c of main function are inaccessible through myAdd or myMull functions.

The variables x, y and z of myMull function are inaccessible through myAdd or main functions, even if myAdd function has the same variables names

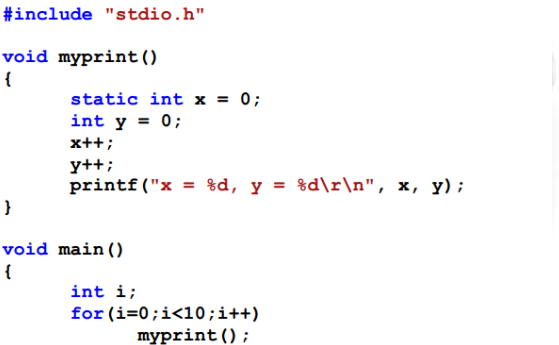
****

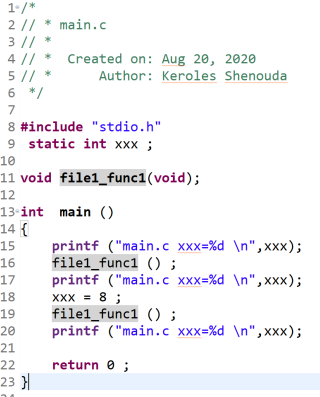
**EX**

The visibility of the automatic variables is limited to the block in which they are defined

The scope of the automatic variables is limited to the block in which they are defined

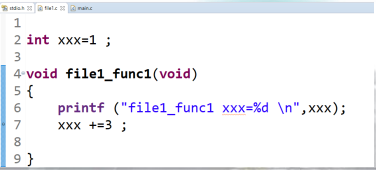
* **Static**
* The variables defined as static specifier can hold their value between the multiple function calls as it is stored on Data Memory.
* Static local variables are visible only to the function or the block in which they are defined
* A same static variable can be declared many times but can be assigned at only one time.
* Default initial value of the static integral variable is 0 otherwise null.
* The visibility of the static global variable is limited to the file in which it has declared.
* The keyword used to define static variable is static



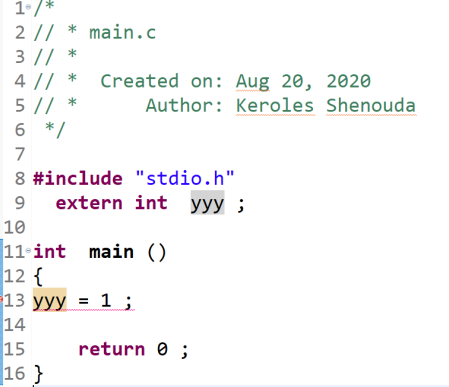
Static variables are defined by the modifier static. Static variables are initialized once in the program life. For example if the variable (x) is defined inside a function, the variable is initialized only at first function call, further function calls do not perform the initialization step, this means that if the variable is modified by any call the modification result remains for the next call. Following example illustrate this idea

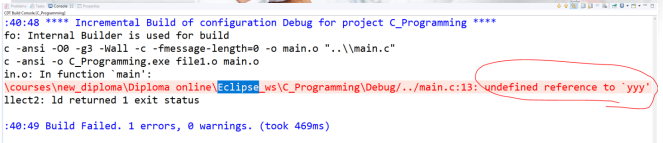
**EX**

The visibility of the static global variable is limited to the file in which it has declared

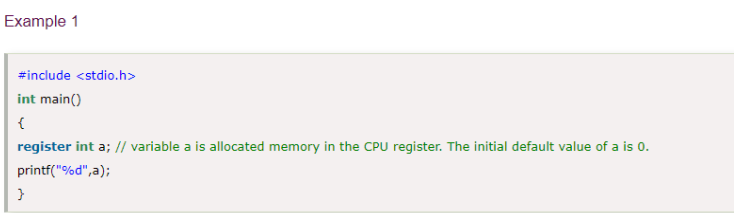


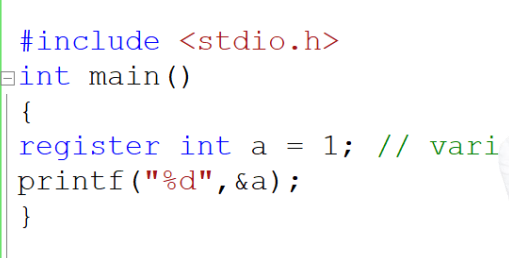
* **Extern**
* The external storage class is used to tell the compiler that the variable defined as extern is declared with an external linkage elsewhere in the program.
* The variables declared as extern are not allocated any memory. It is only declaration and intended to specify that the variable is declared elsewhere in the program.
* The default initial value of external integral type is 0 otherwise null.
* We can only initialize the extern variable globally
* An external variable can be declared many times but can be initialized at only once.
* If a variable is declared as external then the compiler searches for that variable to be initialized somewhere in the program, If it is not, then the compiler will show an error

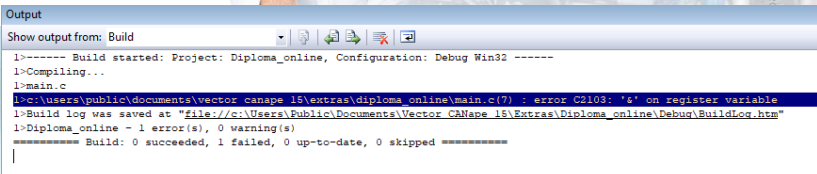




* **Register**
* The variables defined as the register is allocated the memory into the CPU registers depending upon the size of the memory remaining in the CPU.
* We can not dereference the register variables, i.e., we can not use &operator for the register variable.
* The access time of the register variables is faster than the automatic variables.
* The initial default value of the register local variables is 0
* The register keyword is used for the variable which should be stored in the CPU register. However, it is compiler?s choice whether or not; the variables can be stored in the register.
* We can store pointers into the register, i.e., a register can store the address of a variable.
* Static variables can not be stored into the register since we can not use more than one storage specifier for the same variable.









**INLINE ASSEMBLY**

The inline assembler lets us embed assembly-language instructions in our C and C++ source programs without extra assembly and link steps.

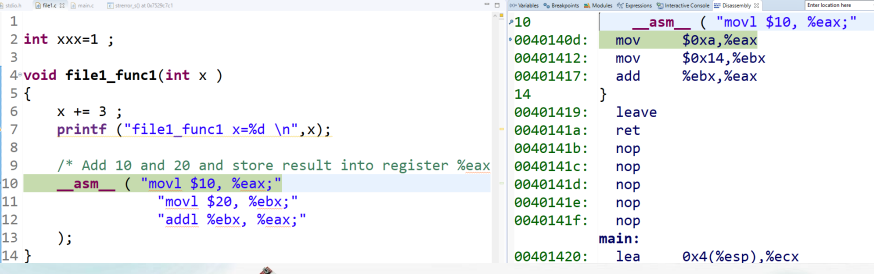
Inline assembly code can use any C or C++ variable or function name that is in scope

**Using GCC**

\_\_asm\_\_("movl %edx, %eax\n\t" "addl $2, %eax\n\t");

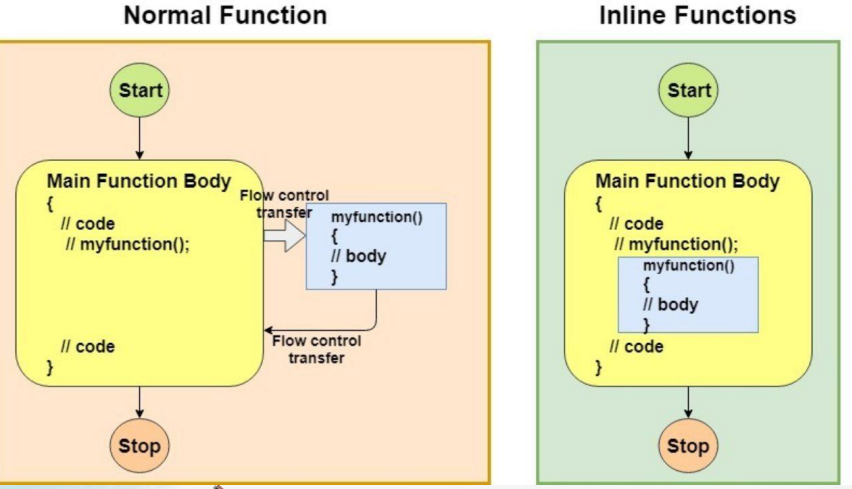
**Using VC++**

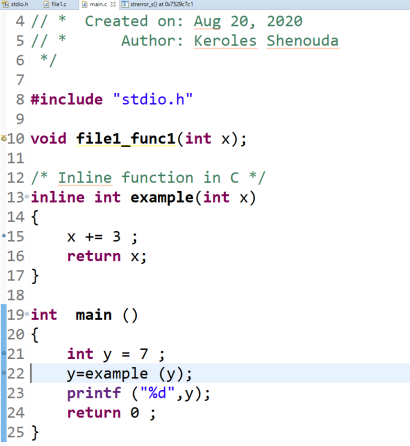
\_\_asm { mov eax, edx add eax, 2 }



**INLINE Function**

Inline Function are those function whose definitions are small and be substituted at the place where its function call is happened. Function substitution is totally compiler choice



**EX**