-SCOPE, VISIBILITY AND LIFETIME OF VARIABLES-

Programming Fundamentals

Some terminologies:

- **Scope**: describes the block/ region in which a variable is **active**(can be accessed).
- **Longevity**: period during which a variable retains its value during entire program execution. (**alive**)
- **Visibility**: accessibility of variable from the memory

Storage Class

- A variable does not retain its value throughout the program.
- The retention of the value depends upon the **storage class** (of the variable).
- A storage class defines the scope (visibility) and life-time of variables and/or functions within a C Program.
- Four types of storage classes:

1. auto
2. register
3. static
4. extern

’auto’ Storage Class

- Default storage class for all the variables declared inside a function or a block.
- Auto variables can be only accessed within the block/function they have been declared and not outside them (which defines their scope).
- Can be accessed outside their scope as well using the concept of pointers (pointing to the exact memory location where the variables resides).
- Assigned a garbage value by default whenever they are declared (if variable is not initialized).

‘register’ Storage Class

- Variables declared as register type have the same functionality as that of the auto variables.
- Only difference: **compiler tries to store these variables in the register of the microprocessor if a free register is available.**
- Use of register variables makes it **much faster** than that of the variables stored in the memory during the runtime of the program.
- If a free register is not available, these are then stored in the memory only. (same as ‘auto’).
- Variables which are to be **accessed very frequently** in a program are declared with the register keyword → improves running time of the program.
- **Cannot obtain the address of a register variable** using pointers.

‘extern’ Storage Class

- Extern storage class: tells us that the **variable is defined elsewhere and not within the same block** where it is used.
- Its **value is assigned in a different block** and this can be overwritten/changed in a different block as well.
- So, it is kind of global variable initialized with a legal value where it is declared in order to be used somewhere else.
- Can be accessed within any function/block.
- Any normal global variable can be made extern as well by placing the ‘extern’ keyword before its declaration/definition in any function/block. It means : we are not initializing a new variable but instead we are using/accessing the global variable only.
- Purpose: Can be accessed across different files which are part of a large program.

‘static’ Storage Class

- Used to **declare static variables**.
- Static variables: **preserves their last value** even after they are out of their scope.
- They are **initialized only once** and exist till the termination of the program.
- **No new memory is allocated** (if we again use static variable) because they are not re-declared.
- Their **scope is local to the function to which they were defined**.
- **Global static variables** can be accessed anywhere in the program.
- By default, they are assigned the **value 0** by the compiler.

Overall Chart:

Storage Class	Storage Area	Default Value	Scope	Lifetime
auto	Stack	Garbage	Within block	End of block
register	CPU Register	Garbage	Within block	End of block
extern	Data Segment	0	Global; across files	Till program ends
static	Data Segment	0	Within block	Till program ends