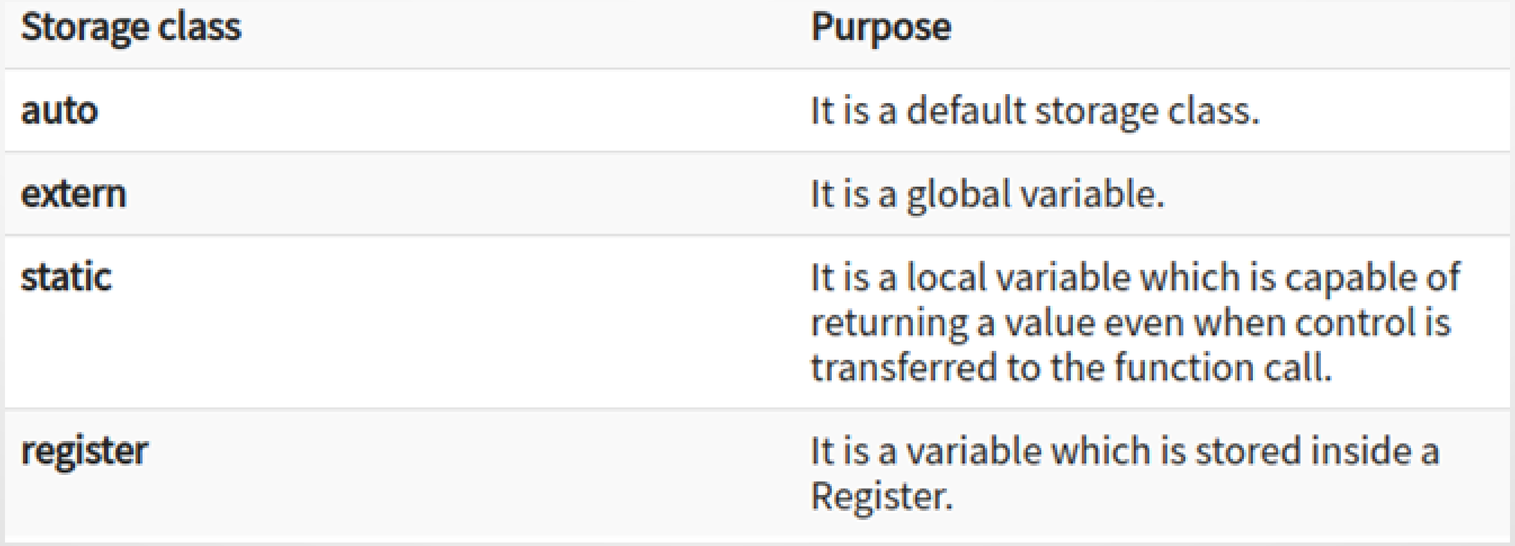
Storage Classes

- A storage class represents the visibility and a location of a variable. It tells from what part of code we can access a variable. A storage class is used to describe the following things:
- The variable scope.
- The location where the variable will be stored.
- The initialized value of a variable.
- A lifetime of a variable.
- Who can access a variable?
- Thus a storage class is used to represent the information about a variable.
- **NOTE**: A variable is not only associated with a data type, its value but also a storage class.

Storage Classes

- **There are total four types of standard storage classes. The table below represents the storage classes in 'C'.**

   1. **auto**
   2. **register**
   3. **Static**
   4. **extern**



Storage class	Purpose
auto	It is a default storage class.
extern	It is a global variable.
static	It is a local variable which is capable of returning a value even when control is transferred to the function call.
register	It is a variable which is stored inside a Register.

AUTO

- The variables defined using auto storage class are called as local variables. Auto stands for automatic storage class.
- A variable is in auto storage class by default if it is not explicitly specified.
- The scope of an auto variable is limited with the particular block only.
- Once the control goes out of the block, the access is destroyed.
- This means only the block in which the auto variable is declared can access it.
- A keyword auto is used to define an auto storage class. By default, an auto variable contains a garbage value.
  - Example, auto int age;

Register

- To have fast access to variables , we can store the variables in the cpu registers.
- C language provides us a storage class called register for the said purpose.
- The scope of register variable is local. They exist as long as the block in which it is declared is active.
- By default register variables assigns garbage value to variables.
- The register storage class when you want to store local variables within functions or blocks in CPU registers instead of RAM to have quick access to these variables. For example, "counters" are a good candidate to be stored in the register.
  - Example: register int age;
- The only difference is that the variables declared using register storage class are stored inside CPU registers instead of a memory. Register has faster access than that of the main memory.

Register

- To have fast access to variables , we can store the variables in the CPU registers.
- Syntax : register data type identifier;
- Ex : register int i =1;

External

- In some applications it may be useful to have data which is accessible from within any block and/or which remains in existence for the entire execution of the program. Such variables are called global variables, and the C language provides storage classes which can meet these requirements; namely, the external.
- External variables may be declared outside any function block in a source code file the same way any other variable is declared; by specifying its type and name. No storage class specifier is used - the position of the declaration within the file indicates external storage class. Memory for such variables is allocated when the program begins execution, and remains allocated until the program terminates. Fo rmost C implementations, every byte of memory allocated for an external variable is initialized to zero.

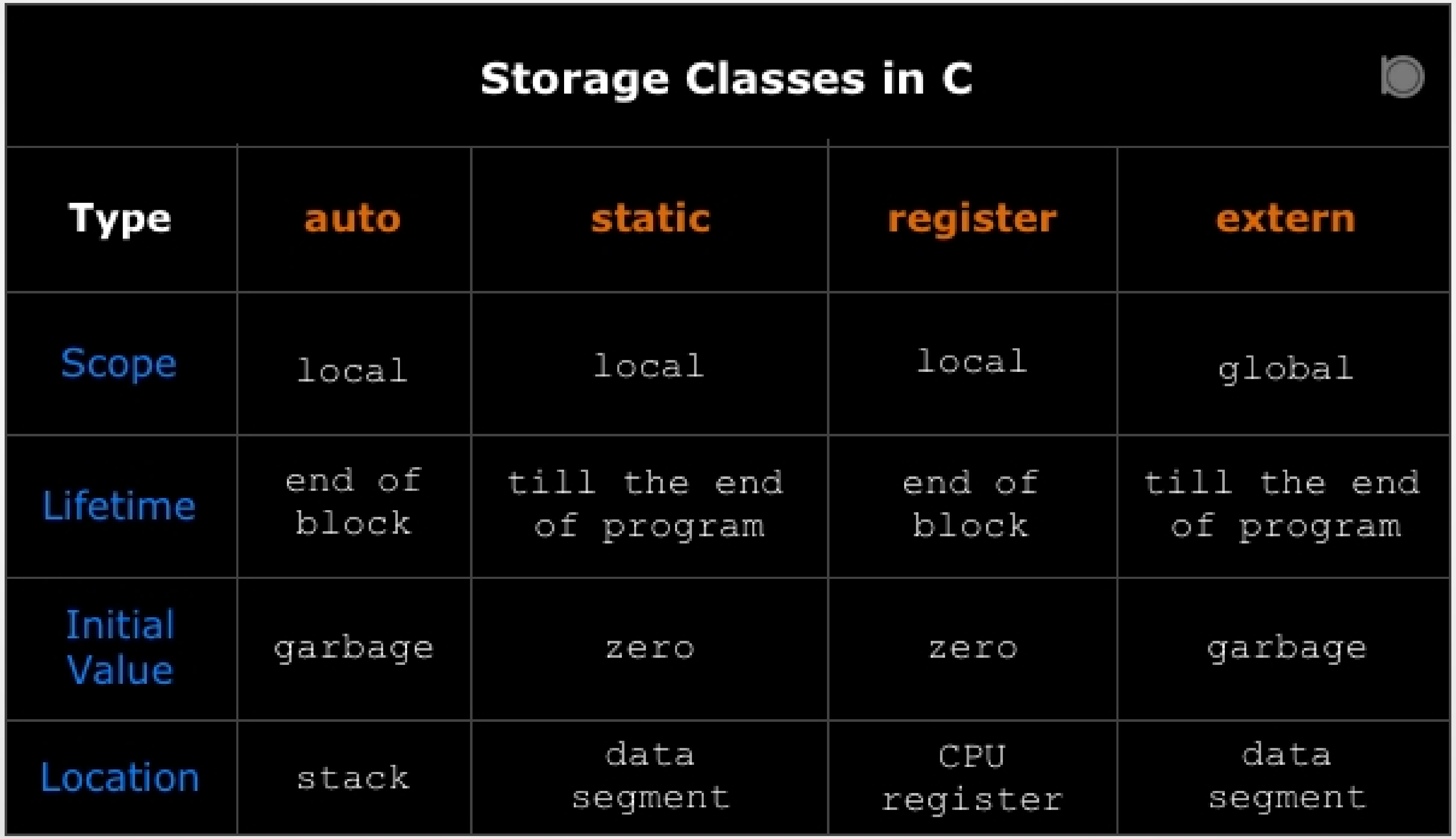
External

- #include<stdio.h>
- int a=10;
- int main()
- {
- ---
- ----
- }

Static

- All the three storage classes create temporary variables in the memory.
- The static storage class variables when defined are made permanent within specified region.
- This variable is stored permanenty in the primary memory and by default , is assigned value zero
- Syantax : static datatype identifier;
- Ex : static int a;

Summery of Storage Classes



Storage Classes in C

Type	auto	static	register	extern
Scope	local	local	local	global
Lifetime	end of block	till the end of program	end of block	till the end of program
Initial Value	garbage	zero	zero	garbage
Location	stack	data segment	CPU register	data segment