Storage Class - Storage Classes associated with variables: automatic, static,

external and register.

teach me all this with appropriate points, differences to note, syntax and an example

Here's a clear guide to the four main storage classes in C: **automatic (auto), static, external (extern), and register**. Each controls a variable's **scope**, **lifetime**, **default value**, and **storage location**. Below you'll find definitions, differences, syntax, and examples for each.

1. Automatic Storage Class (auto)

- **Keyword:** `auto` (optional; default for local variables)
- **Scope:** Local to the block/function where declared
- **Lifetime:** Created on entry to block, destroyed on exit
- **Default Value:** Garbage (uninitialized)
- **Storage Location:** RAM (stack) [1] [2] [3] [4] [5]

Syntax:

```
auto int x; // usually just 'int x;' is used
```

Example:

```
void func() {
    int a = 5;    // 'a' is auto by default
    auto int b = 10; // explicit 'auto'
    printf("%d %d\n", a, b);
}
```

2. Static Storage Class (static)

- **Keyword:** `static`
- **Scope:**
    - If inside a function: local to the function/block
    - If outside any function: local to the file
- **Lifetime:** Entire program (retains value between function calls)

- **Default Value:** Zero (if not explicitly initialized)
- **Storage Location:** RAM (data segment) [1] [6] [2] [3] [4] [5]

Syntax:

```
static int x = 0;
```

Example:

```
void counter() {
    static int count = 0; // retains value between calls
    count++;
    printf("Count: %d\n", count);
}
```

Calling `counter()` multiple times will print incremented values.

3. External Storage Class (extern)

- **Keyword:** `extern`
- **Scope:** Global (across multiple files)
- **Lifetime:** Entire program
- **Default Value:** Zero
- **Storage Location:** RAM (global data segment) [1] [2] [3] [4] [5]

Syntax:

```
extern int x; // Declaration, not definition
```

Example (across two files):
File1.c

```
int num = 100; // Definition
```

File2.c

```
extern int num; // Declaration
void printNum() {
    printf("%d\n", num);
}
```

This allows sharing variables across files.

4. Register Storage Class (register)

- **Keyword:** `register`
- **Scope:** Local to the block/function where declared
- **Lifetime:** Created on entry to block, destroyed on exit
- **Default Value:** Garbage (uninitialized)
- **Storage Location:** CPU register (if available; otherwise RAM)[1] [2] [3] [4] [5]

Syntax:

```
register int x;
```

Example:

```
void fastLoop() {
    register int i;
    for(i = 0; i < 1000; i++) {
        // Fast access variable
    }
}
```

Note: You cannot use the address-of operator (`&`) on a register variable.

Summary Table

Storage Class	Keyword	Scope	Lifetime	Default Value	Storage Location
Automatic	auto	Local/block	Block/function	Garbage	RAM (stack)
Static	static	Local/file	Entire program	Zero	RAM (data)
External	extern	Global	Entire program	Zero	RAM (data)
Register	register	Local/block	Block/function	Garbage	CPU register

Key Differences and Notes

- **auto** is the default for local variables; `auto int x;` is the same as `int x;`[1] [2] [3].
- **static** inside a function keeps its value between calls; outside, limits scope to the file[6] [3].
- **extern** allows sharing variables across files; it only declares, not defines[1] [4] [5].
- **register** requests fast storage in CPU registers; address cannot be taken[1] [4] [5].

Understanding storage classes helps you control variable visibility, lifetime, and memory usage in C programs.

⁂

1. https://www.scholarhat.com/tutorial/c/storage-class-in-c
2. https://www.scaler.com/topics/c/storage-classes-in-c/
3. https://www.shiksha.com/online-courses/articles/storage-classes-in-c/
4. https://www.tutorialspoint.com/cprogramming/c_storage_classes.htm
5. https://www.lix.polytechnique.fr/~liberti/public/computing/prog/c/C/CONCEPT/storage_class.html
6. https://codedamn.com/news/c/auto-vs-static-storage-class-in-c