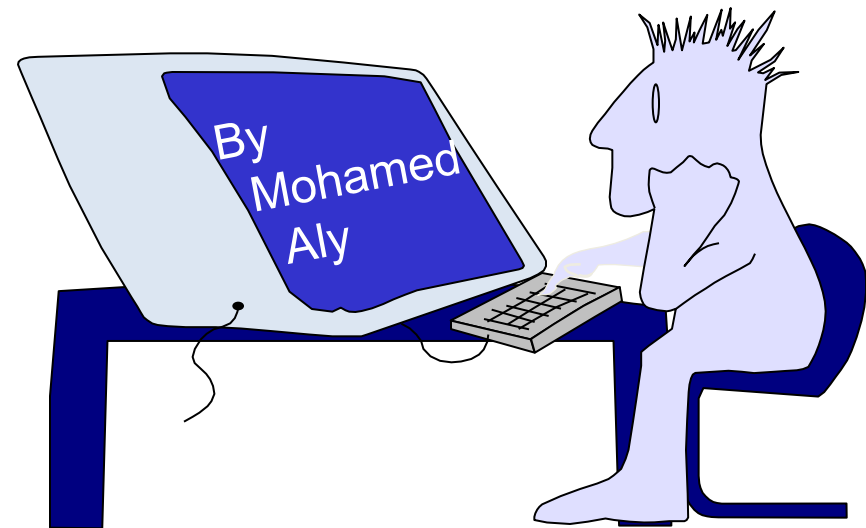


# *Embedded C*

## *C Programming Part 6*



# *Storage classes and scope rules*

- Visibility vs Lifetime
- Variables declared within a function
  - Arguments and Local Variables
  - Visible in remainder of function
  - Lifetime = Function Call
  - Each call obtains a new set of variables
    - Recursive calls too
  - C “internals”
- Variables declared outside any function
  - Visible in remainder of file (!!!)
    - include .h file
    - extern vs static
  - Lifetime = Whole Program
  - C “externals”
- Malloc'd objects

```
int ave(int A, int B)
{
    int C = (A + B)/2;
    return C;
}
```

```
int count = 0;
int fib(int n)
{
    count++;
    if (n <= 2) return 1;
    return fib(n-1)+fib(n-2);
}
```

# *Storage classes and scope rules(Cont.)*

## *Scope rules*

- File scope
  - Identifier defined outside function, known in all functions
  - Used for global variables, function definitions, function prototypes
- Function scope
  - Can only be referenced inside a function body
  - Used only for labels (start:, case: , etc.)
- Block scope
  - Identifier declared inside a block
    - Block scope begins at definition, ends at right brace
  - Used for variables, function parameters (local variables of function)
  - Outer blocks "hidden" from inner blocks if there is a variable with the same name in the inner block
- Function prototype scope
  - Used for identifiers in parameter list

```
int ave(int A, int B)
{
    int C = (A + B)/2;
    {
        int C = (A+B)/3;
    }
    return C;
}
```

# *Storage classes and scope rules(Cont.)*

## ***Storage classes***

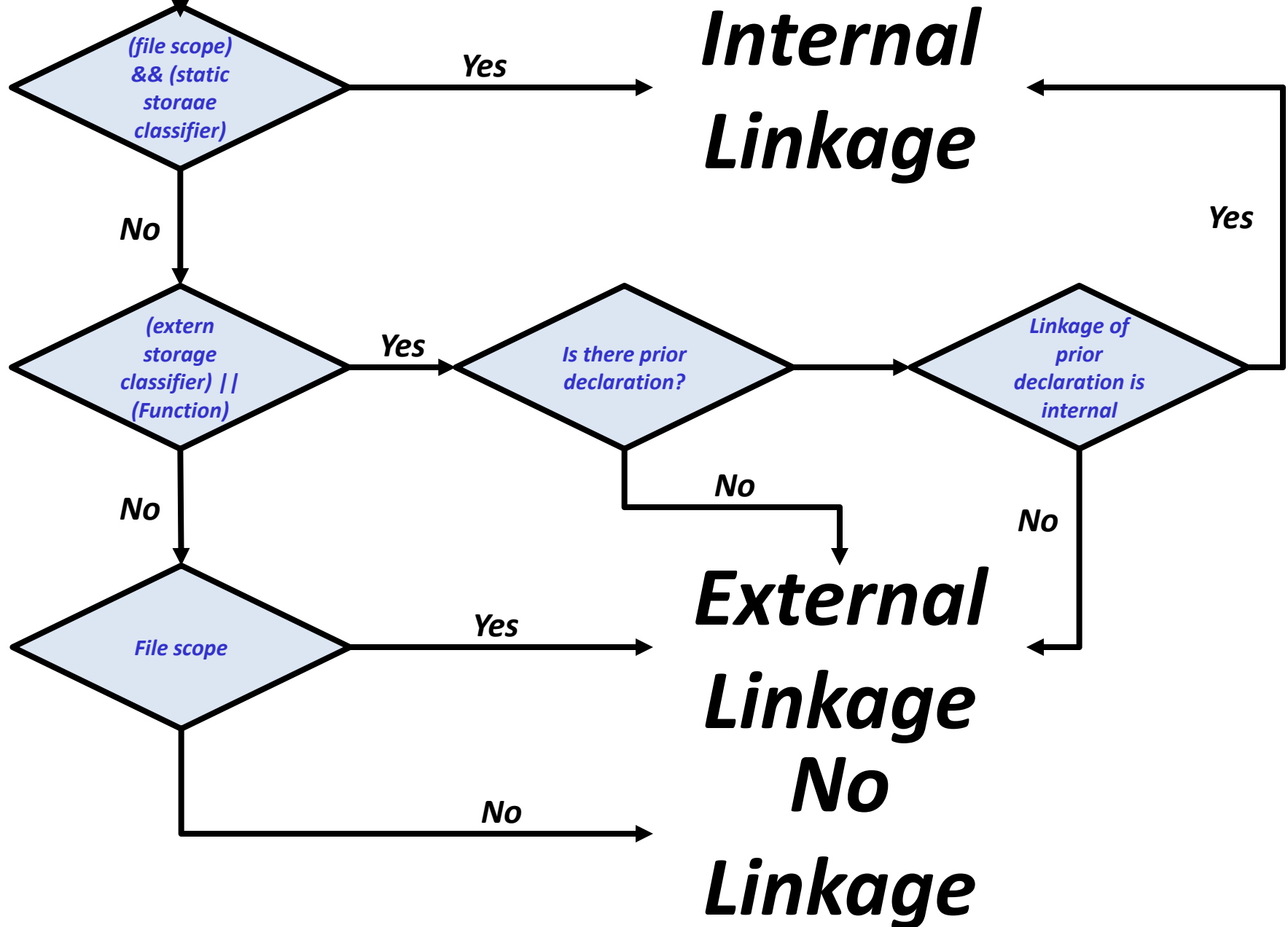
- Automatic storage
  - Object created and destroyed within its block
  - **auto**: default for local variables  
**auto double x, y;**
  - **register**: tries to put variable into high-speed registers
    - Can only be used for automatic variables  
**register int counter = 1;**
    - Can not get address for registered variable
    - Things we'd like to store in registers:
      - Frequently used variables
      - Function parameters
      - Function return values

# *Storage classes and scope rules(Cont.)*

## *Storage classes(Cont.)*

- Static storage
  - Variables exist for entire program execution
  - Default value of zero
  - **static**: local variable or global variable/function
    - For local variables defined in functions.
      - Keep value after function ends
      - Only known in their own function
    - For global variable/function
      - Specifies that variables can only be used in the file in which they are defined
  - **extern**: default for functions
    - Known in any function
    - States that the variable is defined in another file

# *Storage classes, scope rules and linking*



*Thanks*

*Embedded C*