Pointers :→

1. Void Pointer → Type Casted to any particular data type.
2. Data Type Pointer → Points to a particular type of data.
3. Null Pointer → Points to null.
4. Dangling Pointer → Pointing to an address which has been deleted or in a different scope.

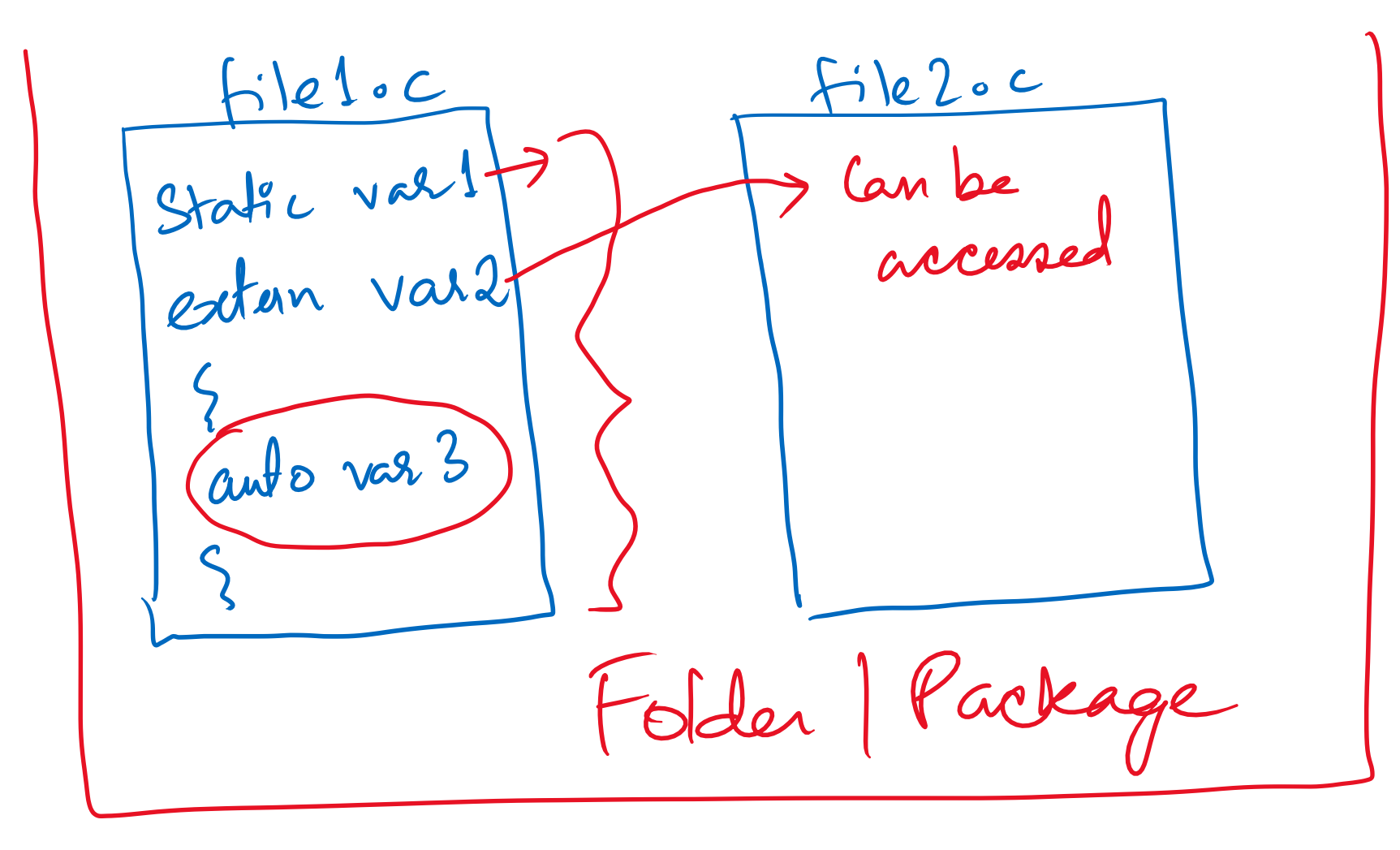
```
function ( )
   x = 10          [2000]
  (2000)           x=10
returning &x;

main ( )              (2000)
int* ptr = function();
   print ("%d", *ptr);   =10
```

Storage Classes:

Storage Classes	Scope	Lifetime
(i) Static	Global	File
(ii) extern	Global	Folder | Package
(iii) register	RAM	RAM Memory
(iv) auto	Local	{ Block Scope }

register → No Pointers Allowed.



Collection of Homogeneous Data : Array

* Collection of Integers
* Collection of Decimal Values
* Collection of Names
* Collection of Characters
* Zero Based Indexing : arr

  Elements: | 9 | 6 | 5 | 2 | 1 | 0 |

  Indexes: 0 1 2 3 4 5
* Values are accessed by indexes.

Note : The name of the array always points to the address of the first element of the array. Therefore the array name is also called "array pointer".

arr = | 2 | 6 | 8 | 5 | → [2000] address

print (arr) → 2000

TCS | Accenture Placement Drive Question:

Given an array of only zeroes, ones & twos sort the array in ascending / descending order without using any sorting algorithm.

I/p → | 2 | 1 | 2 | 0 | 1 |

O/p → | 0 | 1 | 1 | 2 | 2 |

| 2 | 1 | 2 | 1 | 0 | → I/p   index = 0

c0 → ~~1~~ 0   0 1 1 2 2 | c0 - - |

c1 → 2-1-0

c2 → 2 | 0 | 1 | 1 | | |

```
(While (c0>0)
 → While(c1>0)
```

Searching & Sorting Algorithms :

1. Linear Search
2. Binary Search
3. Binary Search using Recursion
4. Jump Search] 6LPA
5. Interpolation Search]

Time Complexity

Comparison Algos:
1. Bubble Sort
2. Insertion Sort
3. Selection Sort
4. Merge Sort
5. Quick Sort
6. Shell Sort
7. Heap Sort

Non Comparison Algos:
8. Count Sort
9. Radix Sort

Binary Search :

Key=12   Key = 6

1. Sorted Array
2. Mid Value

| 3 | 6 | 9 | 10 | 12 | 15 |

0 (s) 1 2 (mid) 3 4 5 (e)

(i) if arr[mid] == key
   return mid;
(ii) if arr[mid] < Key
   S = mid +1;
(iii) if arr [mid] > key → e = m-1

$$mid = \frac{s+e}{2} = \frac{5}{2} = 2$$

$$m = \frac{s+e}{2}$$ (M× M×)

N → N/2 | N/2 → N/4 | N/4 | N/4 | N/4 → $\frac{N}{2^2}$ → $\frac{N}{2^k}$ Log N

$$\frac{N}{2^k} \rightarrow$$

$$K = \log_2 N$$

$$O(K = \log N)$$

$$mid = s + \frac{e-s}{2}$$ Optimal Formula

$$= \frac{2s + e - s}{2}$$

$$= \frac{s+e}{2}$$

* Binary Search Important Questions:

* Search in a 2D Matrix.
* House Robbery I, II, III.
* Book Allocation
* Aggressive Cows
* Square Root of a number using BS.