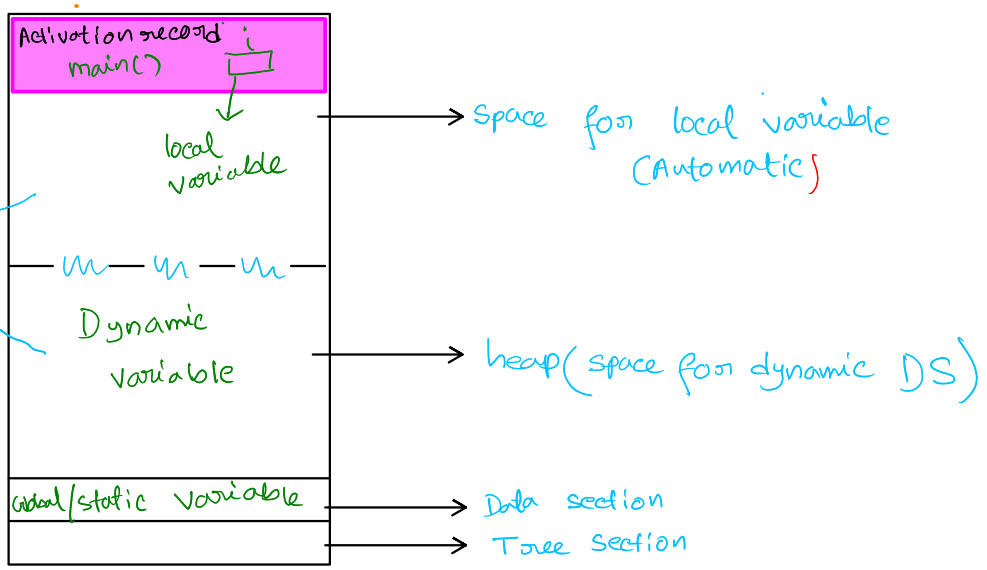
This parts are not fixed if there is more local variable than dynamic ones then space of local variable will have more → If the dynamic variables are more then space of dynamic variable will have more.



Storage classes: Register, extern & static are called storage classes.

Example 1:

```
int main() {
    register int i = 10;
    int *a = &i;
    printf("%d", *a);
    return 0;
}
```

output:

May or May not run

register before a variable declaration saves the variable in the register not in typical activation record.

So if the compiler listen it will save it in the register

But if the compiler does n't support and doesn't listen it will save it in typical activation records.

So, later if we try to access the address of the variable

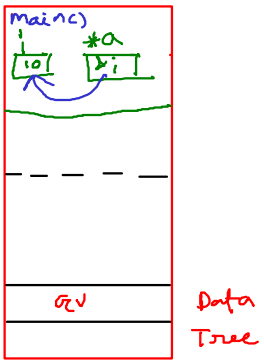
→ i) if the variable is saved in register we will not get any address

ii) if the variable saved in typical activation records we will get the address of the variable

```
int main() {
    int i = 10;
    register int *a = &i;
    printf("%d", *a);
    return 0;
}
```

`int i = 10;` → is saved in activation record

output = 10



2. STATIC: A variable deepration with static storage class has same properties as GLOBAL variable.

i) static & global variable are by default initialised as 0 whereas local or automatic variable has garbage value (any random value) unless they are not intialised as 0.

ii) STATIC or GLOBAL Variable is intialised only once, and even after it is intialised with different value it retains its past value. where as local/automatic variable can be intialised as many time we want.

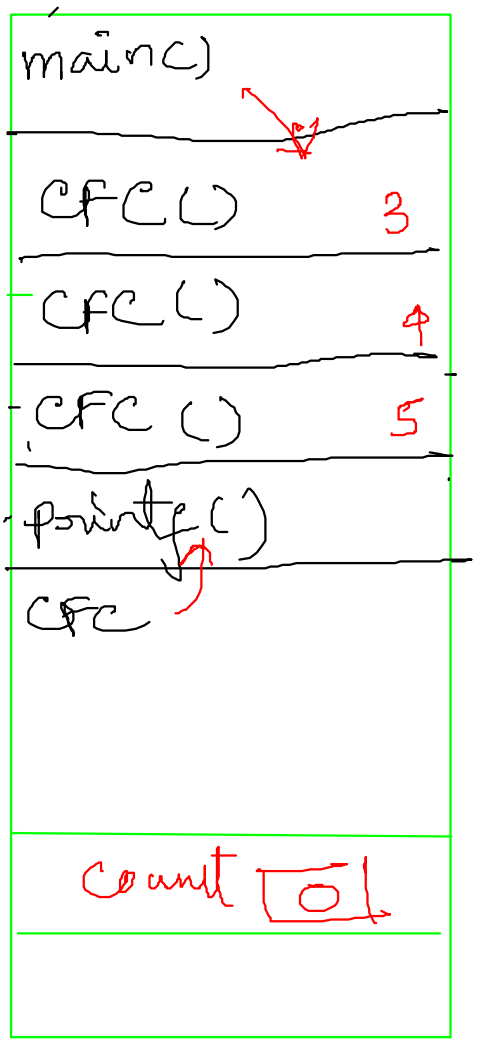
Example.

```
int countFunctionCall(void)
{
    static int count;
    return ++count;
}

int main(){
1
2   countFunctionCall();
3   countFunctionCall();
4   countFunctionCall();
5   printf("%d times function is called", countFunctionCall());
6   return 0;
7 }
```

Here declared the function but didn't catch what it returns → we can do that but its useless

but in the last one we catch the return value of the function.



count = 0

count = 1

count = 2

count = 3

count = 4

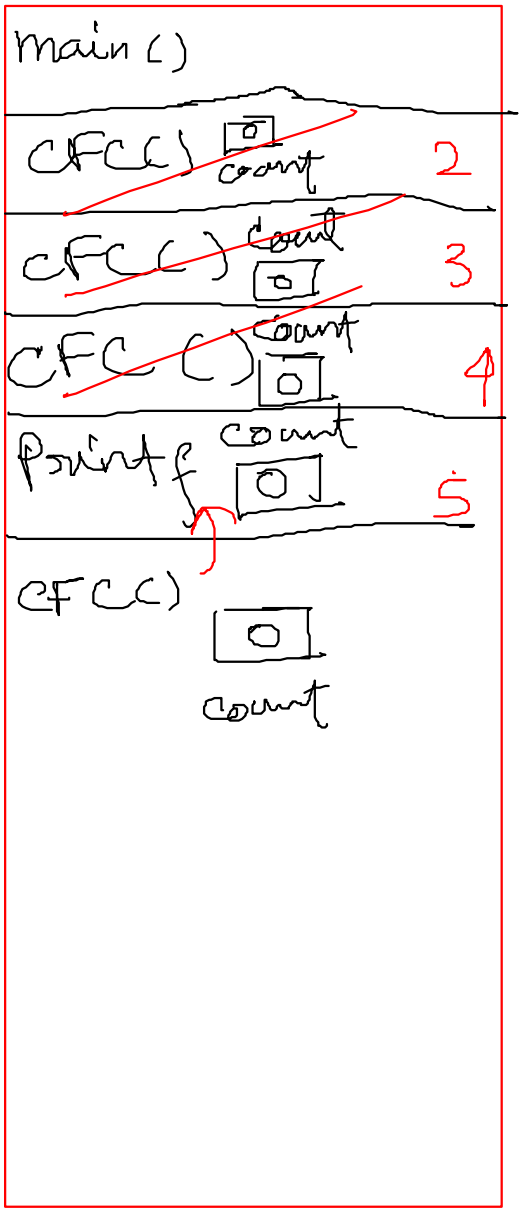
output

count = 4

we could do this b/c count was a static variable, if it was a local variable the the output will be 1 instead 4 b/c local variable count will be initialised as 0 every time the function is called

```
int CountFunCall(void) {
    int count=0;
    return ++count;
}

int main() {
1. countfuncall();
2. countfuncall();
3. countfuncall();
4. printf("%d is the count of fun call", countfuncall());
5 return 0;
}
```



count = 1 not catched

"

"

count = 1

output

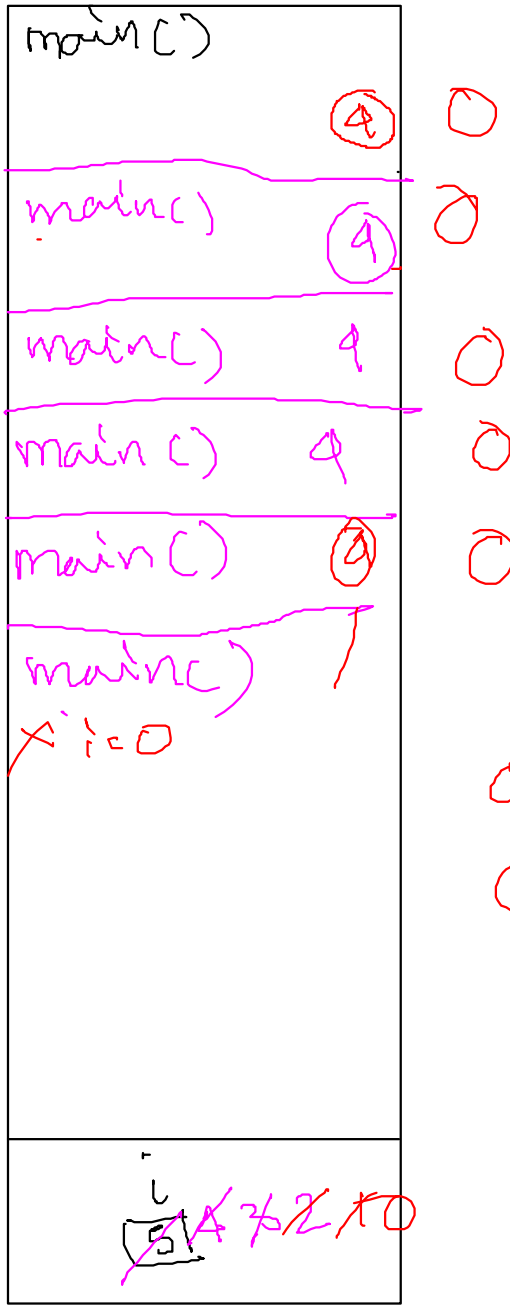
count = 1

```
#include <stdio.h>
int main() {
1   static int i = 5;
2   if (--i) {
3       main();
4       printf("%d ", i);
    }
}
```

if >0 i.e True i=4, 3, 2

output:

4 3 2



output

0 0 0 0 0