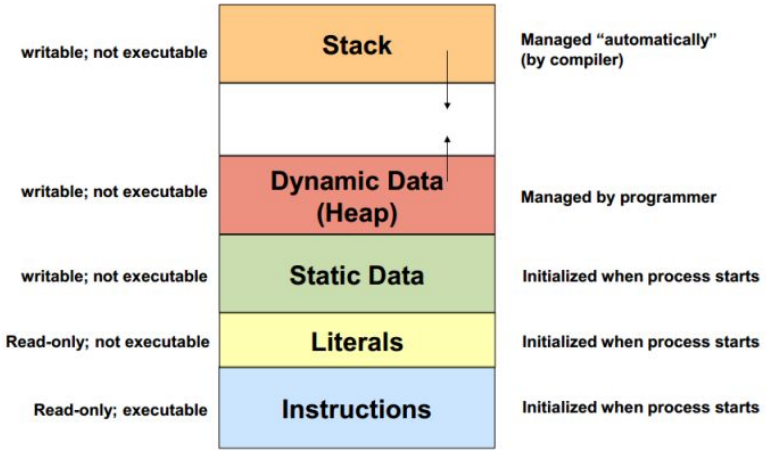
C Recap

Σέργιος - Ανέστης Κεφαλίδης
Κωνσταντίνος Νικολέτος
Κώστας Πλας

Memory

Memory is allocated in 3 ways:

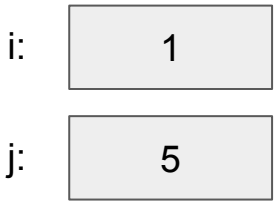
- Static
  - For static or global variables. Space is allocated once, when your program is started and remains reserved till the end of the program. These variables are stored in the data segment.
- Automatic
  - For local variables and function arguments. These variables are stored in the **stack**.
- Dynamic
  - By calling *malloc*. *malloc* allocates memory in the **heap**.



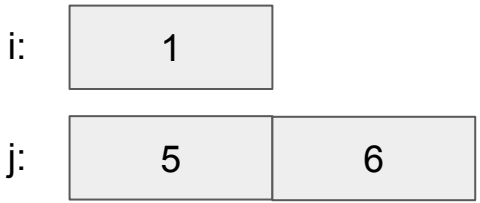
Memory: Automatic allocation

- Memory is allocated and deallocated automatically (handled by the compiler).
- Memory is deallocated when the variable goes out of scope.
- Parameters are also local variables, **copies of the arguments given**.

```
void foo(void)
{
  int i = 1;
  int j = 5;
}
```



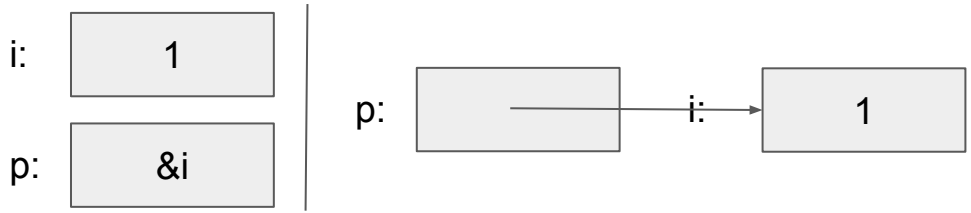

```
void foo(int i)
{
  int j[2] = {5, 6};
}
…
foo (1);
```



Memory: Pointers

- Pointers are normal variables, allocated and deallocated as all other variables.
- Pointers hold memory addresses.
  - We can get the memory address of variable via the *&* operator.
  - We can access the content of the memory address stored in the pointer via the * operator.

```
void foo(void)
{
  int i = 1;
  int* p = &i;
}
```



Memory: Dynamic allocation

- Done by calling *malloc*.
- *malloc* returns a pointer to the address of the allocated memory.
  - The allocated memory resides in the heap.
- Memory allocated in this manner must be deallocated manually via *free*.
  - *free* does not modify the content of the variable (dangling pointer).

```
int main(void)
{
  …
  int* p = malloc(sizeof(int));
  *p = 2;
  // memory stored in p now holds 2
  …
  free(p);
  // p now contains an invalid address
  p = NULL;
  …
}
```

Structs

- A way to store relevant pieces of data together in an orderly manner.
- A collection of variables grouped together under a single name.
  - Each variable in a structure is called a **member**.
- Structs are allocated like any other type.
  - Members are allocated sequentially in memory.

```
struct point {
  int x;
  int y;
}

int main(void) {
  …
  struct point s;
  …
}
```