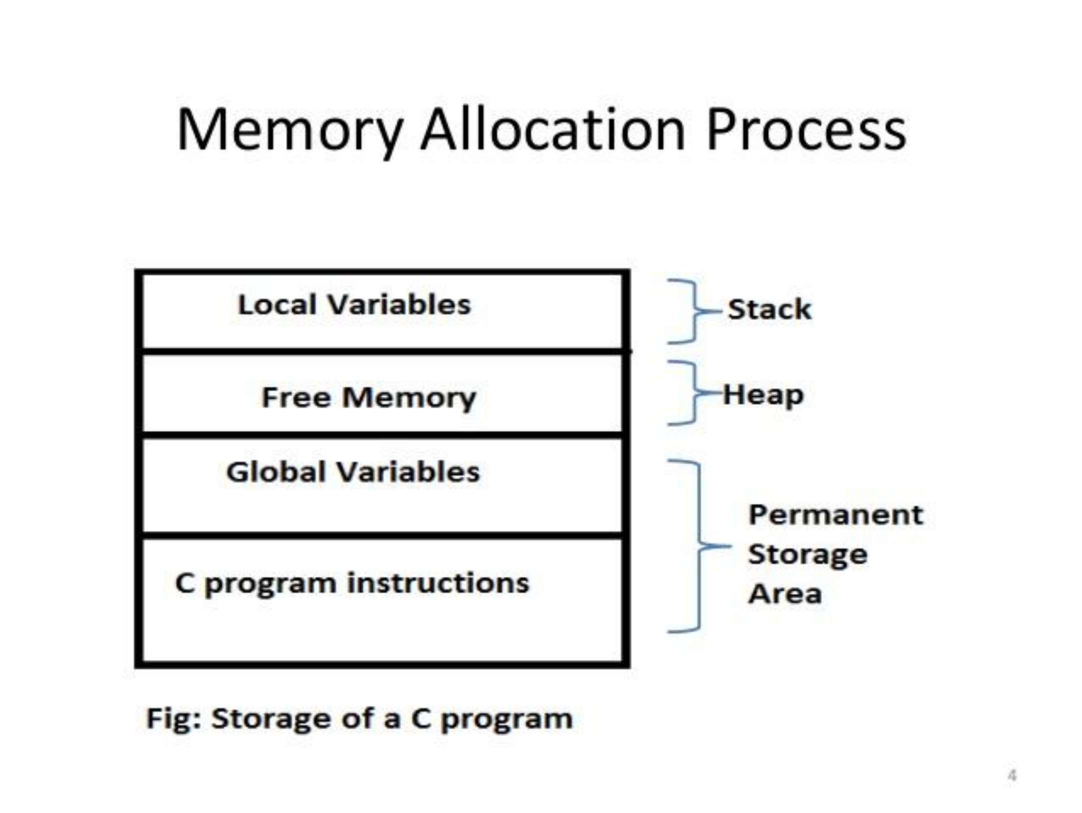
Advanced Topics in Software Engineering

Ahmed Eldably
CODE University of Applied Sciences

Memory Management



Memory Allocation Process

Local Variables

Free Memory

Global Variables

C program instructions

Stack

Heap

Permanent
Storage
Area

Fig: Storage of a C program

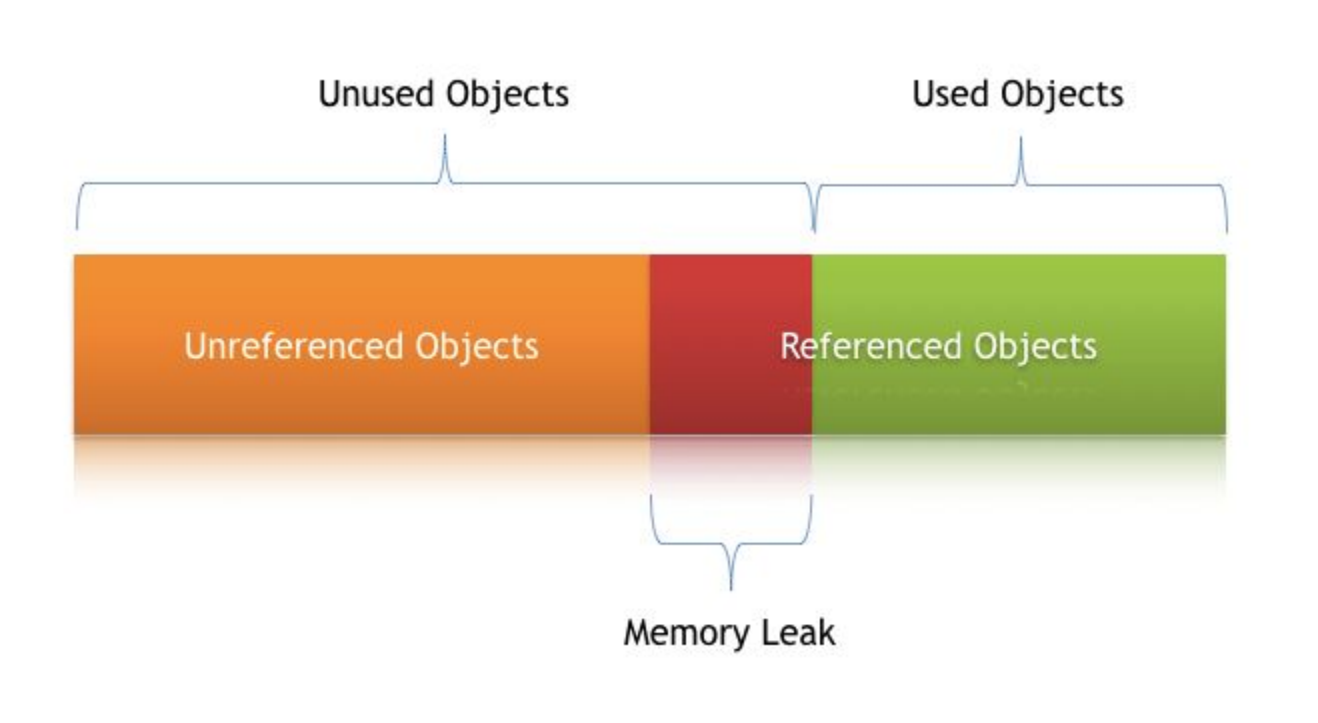
Dynamic Memory Allocation

```
// get an integer from the user
int x = get_int();

// array of floats on the stack
float stack_array[x];

// array of floats on the heap
float* heap_array = malloc(x * sizeof(float));
```

You must free the memory used.



Dynamic Memory Allocation

- Three golden rules:

1. Every block of memory that you `malloc()` must subsequently be `free()`d.

2. Only memory that you `malloc()` should be `free()`d.

3. Do not `free()` a block of memory more than once.

Generic Programming

```
public class MyClass {
    public int[] SuperSort(int[] array) {
            # sorting happens here
        return sorted_array
    }
}
```

Q1: What if I want to sort strings?

Q2: What if I want to sort objects?

Compiler vs Interpreter

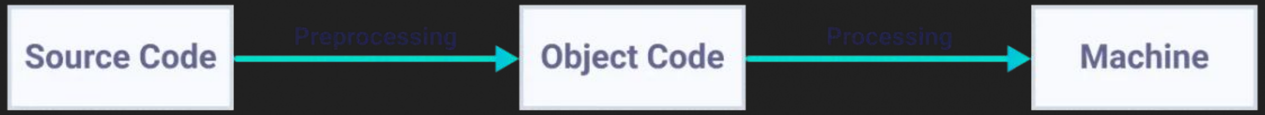


Figure: Compiler

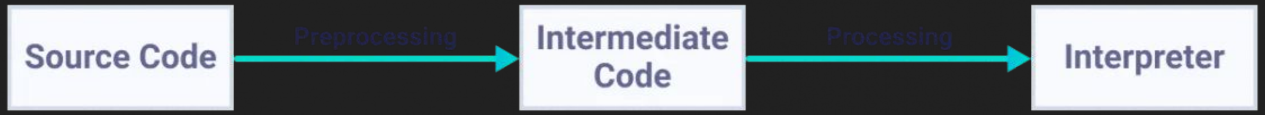


Figure: Interpreter

Lexical Analysis

- First step of compiler (Scanner).
- Converts a high level input into sequence of tokens.

```
int main() {
        // 2 variables
        int a, b;
        a = 10
        return 0;
}
```

All valid tokens are: ‘int’, ‘main’, ‘(‘, ‘)’, … etc # no comments are considered tokens

Parsing

- Converted formatted text into a data structure.
- For example a csv file parsing can result in list of list of values.

```
import csv

wit open(“my_file.csv”, “r”) as csv_file:

    reader = csv.reader(csv_file)

    for row in reader:
        print(row)
```

How parsing works

Input shape:

Family Name, Given Name
Eldably, Ahmed
Fracassi, Fabio

1. Scanning: converting a stream of characters into tokens.
2. Syntactic Analysis: analyses the structure formed to keep token in order as their positions. It also validates the right data structure for the entry

Output shape

{“Family Name” : [Eldably, Fracassi], “Given Name” : [Ahmed, Fabio] }

Code Optimization

```
do {
        x = 5
        y = y + x
}  while(y<100)
```

to:

```
x = 5
do {
        y = y + x
}while(y <100>
```

Code generation

- The process by which some automated tool is used to turn higher level input into lower level (less abstract)

References:

1. CS50 Section on Dynamic Memory Allocation
2. https://www.geeksforgeeks.org/introduction-of-lexical-analysis/
3. https://medium.com/the-mighty-programmer/what-is-parsing-4012f997d265
4. https://www.tutorialspoint.com/compiler_design/compiler_design_code_optimization.htm