To start with, a random number is generated to determine philosopher’s arrival length of time (between 1-10 sec), whenever it arrives putting a plate animation is iterated.

Furthermore, to make them wait each other, a **barrier** is applied via semaphores for each. When a philosopher arrives, s/he releases his/her semaphore by number of total philosophers and acquires others’ semaphores. By this way, it is guaranteed that no thread shall pass the start dining part without everyone arrived.

In dining part there is 3 function and 1 part which are used:

* First part, indicates the thinking action, within the time between 0-10 second (randomly generated)
* After thinking, philosopher becomes hungry and tries to eat spaghetti in front of him/her with the left and right forks. To manage that, function called “check” and semaphore of that philosopher is used. In function “*start\_eat*”, thread tries “*check*” function and if check function does not make semaphore wake up, it is blocked until neighbours’ call.
* Whenever semaphore of philosopher is woken up, it starts to eat (3 secs).
* Finally, it is time to leave forks away via “*stop\_eat*” function and goes back the status of thinking. Moreover, the mutex guarantees that no other threads doing something with forks at the same time (**mutual exclusion**). It also invokes the check function for neighbouring philosopher once again.