Project 1

Fundamentals of programming II

Gússem Yahia-Cheikh and Sara Díez

April 2025

Strategy and Steps

We decided to divide the work by the three core functionalities provided in the project description:

- **Task 1: Package Management**
- **Task 2: Package Classification**
- **Task 3: Shopping simulation**

This divsion helped us to work modularly and focus on each part before combining them toghether.

Argument Management

We started with the agrument validation. The function `void CheckArguments (int argc, char *argv)` ensures that the correct number of arguments is passed and that the second argument is a positive integrer. If valid, it's stored in `EventNumbers` , which controls the number of simulated events.

Event Loop Setup

We then implemented the `SimulationLoop(int EventNumbers)` function, which works as the heart of the simulator:

1. Loops over the number of events
2. Generates a random event type (0,1 or 2)
3. On each iteration we delegate it corresponding type:
    1. Type 0: Package sorting
    2. Type 1: Package classification
    3. Type 2: Shopping
4. And finally simulating the time passing after each event (`UpdateShoppingQueue()`)

This ensures that exactly one time unit passes per event.

Task 1: Package management

Now let's address the package management block, here worked with sorted linked lists ordered by `suppliers` (taking in count that the `RobotPackage` contains a `supplier`, `id` and `year`). We have three functions working in this block: `PrintRobotPackages()`, `SearchRobotPackage()` POSAR ALGO MAS

Task 2: Package Classification (Stacks)

For this task, we implemented three stacks representing small, medium, and large packages.

Each `Package` has a randomly assigned type and color. When a package arrives, the system decides which stack it belongs to and pushes it there.

Each time a package is classified, we check whether the stack has reached its `MAX_CAPACITY`. If it has, the robot simulates transporting all packages by removing (freeing) all nodes in the stack.

Task 3: Shopping (Queue and Time)

This system manages a shopping queue. Each robot is created with a shopping list (a number of items to buy) and added to the queue.

Time is simulated through the event loop: one unit of time passes for each event (sorting, classification, or shopping). We track how many items each robot still has to buy. When a robot's count reaches zero, it is removed from the queue (and memory is freed).

Simulation Design

The simulation is a key piece in this project. Each iteration of the event loop represents one time unit, this is enforced consistently calling `UpdateShoppingQueue()` after handling each event. As a result, robots shop one item per event until their list is complete.

Problems and criticisms

One of the biggest challenges was managing dynamic memory correctly. It was easy to forget freeing certain structures, especially when dealing with stacks and queues.

Another difficulty was simulating the shopping time using event counts. We had to ensure that time progressed accurately and that no robot was skipped or served out of order.

Finally, handling the string operations and ordering in the package management system required careful use of `strcmp()` and array boundaries.

Output Sampling

Terminal copy-paste:

```
gussemyahia-cheikh@Gussems-MacBook-Pro FP2LAB1 % ./file.exe 100
Starting...
Robot Packages (Sorted List):
Supplier: CISCO, ID: CIS-X1-GRE-ME, Year: 2004
Supplier: CISCO, ID: CIS-X1-GRE-ME, Year: 1986
Supplier: CISCO, ID: CIS-X1-GRE-ME, Year: 1998
Supplier: CISCO, ID: CIS-X1-GRE-ME, Year: 2012
```

```
Supplier: CISCO, ID: CIS-X1-GRE-ME, Year: 2012
Supplier: CISCO, ID: CIS-X1-GRE-ME, Year: 1986
Supplier: CISCO, ID: CIS-X1-GRE-ME, Year: 2002
Supplier: DELL, ID: DEL-DX-PLA-LAR, Year: 2015
Supplier: DELL, ID: DEL-DX-PLA-LAR, Year: 1983
Supplier: DELL, ID: DEL-DX-PLA-LAR, Year: 1995
Supplier: DELL, ID: DEL-DX-PLA-LAR, Year: 2018
Supplier: IBM, ID: IBM-OX-PLA-LA, Year: 2007
Supplier: IBM, ID: IBM-OX-PLA-LA, Year: 1992
Supplier: IBM, ID: IBM-OX-PLA-LA, Year: 1989
Supplier: IBM, ID: IBM-OX-PLA-LA, Year: 1986
Supplier: IBM, ID: IBM-OX-PLA-LA, Year: 1981
Supplier: IBM, ID: IBM-OX-PLA-LA, Year: 2018
Supplier: Intel, ID: INT-EX-ORA-ME, Year: 2016
Supplier: Intel, ID: INT-EX-ORA-ME, Year: 1986
Supplier: Intel, ID: INT-EX-ORA-ME, Year: 1990
Supplier: Intel, ID: INT-EX-ORA-ME, Year: 1985
Supplier: Intel, ID: INT-EX-ORA-ME, Year: 1994
Supplier: Intel, ID: INT-EX-ORA-ME, Year: 2010
Supplier: Lenovo, ID: LEN-IN-SIL-LA, Year: 1985
Supplier: Lenovo, ID: LEN-IN-SIL-LA, Year: 2018
Supplier: Philips, ID: PHI-TS-BLU-SM, Year: 2011
Supplier: Philips, ID: PHI-TS-BLU-SM, Year: 2008
Supplier: Philips, ID: PHI-TS-BLU-SM, Year: 2001
Supplier: SONY, ID: SON-XR-BRO-SM, Year: 2007
Supplier: SONY, ID: SON-XR-BRO-SM, Year: 1988
Supplier: SONY, ID: SON-XR-BRO-SM, Year: 1984
Supplier: SONY, ID: SON-XR-BRO-SM, Year: 2011
Supplier: SONY, ID: SON-XR-BRO-SM, Year: 1980
Supplier: Samsung, ID: SAM-VK-DBL-LA, Year: 1986
Supplier: Samsung, ID: SAM-VK-DBL-LA, Year: 2013
Supplier: Samsung, ID: SAM-VK-DBL-LA, Year: 1985
Supplier: Samsung, ID: SAM-VK-DBL-LA, Year: 2015

Stack 0:
Package Type: 0, Color: beige
Package Type: 0, Color: yellow

Stack 1:
Package Type: 1, Color: beige

Stack 2:
Package Type: 2, Color: green
Package Type: 2, Color: white


STATISTICS WHEN CLEANING THE SIMULATION:
Removing packages...
3 packages have been removed.
Cleaning all stacks of packages...
12 packages have been removed.
Cleaning shopping queue...
43 robots have been removed.
```

```
No robots remain in the shopping queue.
```