Degree: Artificial Intelligence
Subject: Fundamentals of Programming 2
Practical project 1

***Abstract*—During this second semester of the 2024-2025 academic year, you will work on 2 practical projects in the Fundamentals of Programming 2 subject. This document presents the first one, which is related to the concepts we've seen up to now. With the double aim of putting into practice the knowledge obtained in this subject and of acquiring independent work habits, this Practical Project 1 will focus on the development of a program in C, which includes different aspects of programming in C, such as program arguments, static and dynamic memory allocation or dynamic data structures.**

You will have 3 sessions in order to deliver a functional version of the program. However, this project will require more hours of dedication, not only the 3 in-present sessions. During the last session, you will also present the work done.

In this document, we introduce a description of the problem you will have to solve. It will be related to a simulation of robots that help us in our day of every day.

***Index Terms*—C programming, dynamic memory, program arguments, linked lists, stacks, queues, simulator based on events, Artificial Intelligence.**

I. Introduction

With the double objective of putting into practice the knowledge obtained in this subject and of acquiring independent work habits, the laboratory practices will focus on the development of a C program that will cover different aspects seen up to now. In this project, we propose you a set of possible milestones (implementation of different functionalities of the program) and we define the deadline in which you will have to deliver the work done.

The planning necessary to achieve these milestones, and even deciding which milestones you want to achieve, is your responsibility.

In Section II we present the description of the proposed program. In Section III we explain you the template code you will be provided with. In Section IV we describe the milestones that the project may have (although they depend on you and your organization) and in Section V we give some recommendations and general orientations. Finally, in Section VI you will find the delivery description and how the project will be assessed.

II. Problem description

Currently, the technology arrives to the point that automatic systems and robots are available to help us at work and at home, as it can be seen on Figure 1. In order to analyze different approaches, check different behaviours and assess their productivity, we have to develop a preliminary version of a program that will simulate an automatic system that collaborates with robots' tasks.

For this practice, we assume that a logistics company requires the automation of different parts of its system seeking to improve productivity and to reduce costs. In this context, we are asked to develop, as specialists in system automation, a prototype of a simulator that would provide 3 types of automated systems: (i) package management, (ii) package classification and (iii) a new service called "directly to customer", that makes purchases directly in stores and delivers them using robots.

For this preliminary version of the simulator, we will work with an automatic system and robots that will help us in: managing packages, classifying packages and going for shopping.

Fig. 1. Example of automatic system and robots.

The simulator will be driven by events. The number of events must be passed to the program as an argument from the command line. We provide you with a template (files `.c` and `.h`) where we included a base for the program, in particular, functions to generate a set of events in order to check the correct functionality of your program.

The simulator generates events (one by one) using the function we provide you. Then, it starts consuming them (event by event). Depending on the generated event type (type of system task), a corresponding automatic system or robot proceeds with the event execution.

A. System task 1: Package management

Functionality The first system to be developed will facilitate the management tasks of packages that are managed in the logistics company that are executed by a special robot. This system must allow the organization of the warehouse, for this initial version, by sorting only one of its shelves. The system, with the help of a robot, will look through the current list of packages, searching for a space to place it. Packages are identified by an identifier and will be sorted by supplier name (Figure 2). The list of packages will be maintained until the end of the program. The system must provide information on the final number of packages in the stock. At the end of the execution of the program, all the existing packages on the shelf (on the list), will be released using the `free()` function.

Fig. 2. Sorting packages.

Data structure

RobotPackage – Each package will contain the following fields: A supplier (a string of characters), An identifier (a string of characters), and the manufacturing year (integer number). Static arrays can be used for the string fields, for example:

```
char supplier[20];
```

Be careful when using arrays of characters in C. Please, see Section V for further information about how to use strings in C.

B. System task 2: Package classification

Functionality This second system will be in charge of classifying the entry of packages, separating the packages that enter considering the different sizes (Figure 3). Initially, there are 3 different types of packages (small, medium and large), and they are separated and placed in a corresponding stack, based on their sizes. In this way, we have 3 types of packages and a separate stack for each of them (i.e., a stack for small packages, a stack for medium packages, and a stack for large packages). The system controls where each package must be placed on and how many packages are in each stack. In the beginning, all stacks are empty.

Each stack is characterized by its maximum capacity. Packages can be put in the stack while this capacity is not reached. If the maximum capacity is reached, the robot can remove all

Fig. 3. Classifying packages.

the packages from the stack and put them for the transport (we don't manage that issue). When the robot removes all the packages from the stack, the stack will be empty again. To control the maximum stack size, we've defined MAX_CAPACITY - a constant value given in a template.

Since the number of packages depends on the number of events generated by our simulator, at the end of the program execution, there may be packages left in the stacks. In this situation, the program provides (prints) information about the final number of packages that are still in the stack and that have not been removed from the stacks to be transported (total number of packages that are remaining in all stacks). After this, these packages will be freed using the `free()` function.

Data structure

Package - Each package has a package type (small, medium and large). Additionally, each package is characterized with a color code (white, green, yellow and beige).

Please, see Section V for further information about enumeration types in C.

C. System task 3: Shopping

Functionality

For system task 3, we will have a set of robots. In this case, each robot will take a list of purchases (i.e. the robot has a number of things to buy) and will go to the supermarket for shopping (see Figure 4. Each robot will go for shopping only once.

Fig. 4. Shopping.

When an event of shopping appears, a new robot is created for this task and a list of purchases is assigned to it (i.e. the

number of things to buy will be assigned to it). The robot goes to the queue (supermarket queue) and is added to the end of the queue. Each robot will wait in the queue to be served if there are more robots before it. When its turn comes (i.e. no other robot is buying), it will be removed from the queue (i.e. released using `free()`) and it will buy all things from the shopping list (i.e. a number of things to buy). After the shopping is finished, it can go back home.

In this case, a simulation of period of time must be developed. We will do it in a very simple way: it will be based on the number of events consumed by the simulator and the number of things to buy.

The robot is added to the queue. If there are more robots already in the queue, it will be added at the end of the queue and it must wait for its turn. The time will be passing depending on the events consumed (events from the main program loop: package management, package classification and shopping). One unit time passes when one event (from the main loop of events) is consumed (for example, one package was classified). Two units of time passes when two events are consumed (for example, one package was sorted, one package was classified), etc.

To give a clearer example: when a robot with ID=4 is generated to buy 3 things, it is added to the queue. If there are no other robot in the queue, it will be directly removed from this queue (i.e. released using `free()`) and served (it will buy its 3 things during 3 events consumed from the main loop (e.g. 1 sorting packages, 1 classifying packages, 1 sorting packages).

If there are other robots in the queue before robot with ID=4 (e.g. robot ID=2 with 2 things to buy, robot ID=3 with 1 thing to buy), robot ID=4 will wait for (2+1=3) 3 events to be served.

You will have to develop a mechanism for counting the events and controling the robot's shopping time. As the number of robots depends on the number of events generated by our simulator, at the end of the program execution, there still might be robots in the shopping queue. In this situation, the program gives information about the final number of robots that are still in the queue (the total number of robots that are still in the shopping queue). After that, these robots will be released using `free()`.

Data structure

Shopping – number of things to buy (integer number) and the identifier of robot that goes for shopping (integer number).

III. Program arguments and template

The program must take as the input argument `Number of events` that must be generated by the program.

Remember that, instead of choosing the "Run Code" option from Visual Studio Code, you can compile and execute the program from the terminal. Just input the following command to compile:

```
gcc -o executable source_file.c
```

and then execute the executable file with the corresponding arguments, for example:

```
program 100
```

If a user introduces a different number of arguments, the program must show an error message and indicate which is the correct format of the program execution.

We provide you with a template for starting the simulator implementation. Please, review it well, analyze and implement different functionalities step by step. You can find 2 source code files:

- `project1.h` - The header file that contains all necessary structures and variables, some of them with certain initial values.
- `template.c` - A template for a source code of the simulator. It includes `project1.h`, and hence you can use all the structures and variables declared there. This file is divided in 5 parts:
  1) General functions for an event generation and argument checking
  2) A set of functions for the package sorting functionalities, in particular the implementation of a function for the random package generation
  3) A set of functions for the package classification functionalities, in particular the implementation of a function for the random package generation
  4) A set of functions for the shopping functionalities, in particular the implementation of a function for the random shopping generation.
  5) A `main()` function that will call a simulator loop.

The example outputs and statistics that we can obtain after execution of the program are presented in the Appendix.

IV. Milestones

The goal of the project is to provide all the functionality of the simulator. You can set the following milestones:

1) Passing required arguments to the program from the command line and managing errors
2) Generating events - using the functions we've provided you in the template, you can generate event by event in your code
3) Consuming events - a loop that consumes events (event by event); depending on the type of event, it calls a corresponding robot
4) Functionality of the robot described in Section II-A System tasks 1 - Sorting packages
5) Functionality of the robot described in Section II-B System tasks 2 - Classifying packages
6) Functionality of the robot described in Section II-C System tasks 3 - Shopping

The deadline for this first deliverable is indicated in the Virtual Campus.

V. Recommendations

Here are some recommendations on how to develop the project:

- Properly organize the project with the classmate of your group.
- Comment your code with helpful comments (mostly for yourself). To provide a useful comment, it is very important that it is accompanied by the date and the author.
- To define enumerations in C, you can use a special enumeration type called `enum`. It is mainly used to assign names to integral constants, the names make a program easy to read and maintain. For example:

```
enum PackageType
{
    small=0,
    medium=1,
    large=2
};
```

- Working with strings in C, you can use arrays of characters, for example: `char array[20];` or using dynamic arrays and allocating/releasing memory with `malloc()/free()`, respectively. Moreover, while doing operations on strings in C, you will have to use specific functions, for example:
  - `strlen()` - calculates the length of a string
  - `strcpy()` - copies a string to another
  - `strcmp()` - compares two strings
  - `strcat()` - concatenates two strings
- Remember to save copies of your work from time to time. You can also use version control tools (there are a few with free options):
  - **Bitbucket** https://bitbucket.org
  - **GitHub** https://github.com/
  - **GitLab** https://about.gitlab.com/

  This way, we can avoid panic attacks and problems due to loss of information, catastrophic decisions, etc.

VI. Delivery and assessment

The project will be done in pairs and you will have 3 'laboratory' sessions. Take into consideration, that this project will require more hours of dedication, not only the 3 in-present sessions. In the last session you will have to explain orally the solution and implementations carried out. After the last session you will have a short period (3 days) to hand in a report or fixing small errors. This means that by the last session you should already have the whole implementation and report almost finished and only check/fix/add any small details or observations you may have and comment them with the professors.

A. Delivery via Virtual Campus

The delivery will be done through the Virtual Campus until the indicated date. It's enough if only one of the members of the group deliver the project. You have to attach: the report in pdf format and the source code (`*.h, *.c`) compressed in one `.zip` file.

TABLE I
WEIGHT OF EACH PRACTICAL PROJECT INTO THE SUBJECT'S FINAL GRADE.

Practical project 1	**15%**
Practical project 2	**15%**

Additionally, a delivery report, in PDF format, must contain the following information:

- The student's full name and NIU
- An explanation of the design decisions and their implementation in the solution.
- A description of the main difficulties encountered for the carrying out the practice.
- An explanation of the behavior observed in the executions of the exercise.

B. Assessment

The final grade of the practical projects represents 30% of the subject's final grade. It is an essential requirement to have a grade of 5 or higher in the practical projects to pass the subject. You will have a total of 2 deliveries (two practical projects) during the semester; and this is the first one. The contribution of each project to the final subject's grade is shown in table I. Remember, that you will have a validation probe during the theoretical exam to validate your knowledge about the project.

The following criteria will be used to calculate the grade of this first project:

- [0,5 points] The management of the arguments passed to the program and consideration of possible users' errors.
- [6,5 points] The 'correct' working of the code. There are 3 functionalities:
  - [2 points] Household tasks 1: Sorting packages
  - [2 points] Household tasks 2: Managing packages
  - [2,5 points] Household tasks 3: Shopping
- [1 point] Code style and comments.
- [2 points] Report
  - [1,5 points] Description of the strategy in an accurate and reasoned way.
  - [0,5 points] Criticism of the problems that arose and the solutions found during the development of the project.

VII. Conclusions

We have already presented you the first of two practical projects you will develop this semester. Please, contact us if you have any doubts or questions: `Ernest.Segui@uab.cat` and/or `Nehir.Sonmez@uab.cat`. And do not hesitate to ask for meetings with us!

Appendix

Final example statistics of the program execution for different configurations

```
program 10
STATISTICS WHEN CLEANING THE SIMULATION:
    Removing packages...
        2 packages have been removed.
    Cleaning all stacks of packages...
        5 packages have been removed
    Cleaning shopping queue...
        0 robots have been removed

program 100
STATISTICS WHEN CLEANING THE SIMULATION:
        Removing packages...
            29 packages have been removed.
        Cleaning all stacks of packages...
            2 packages have been removed
        Cleaning shopping queue...
            9 robots have been removed

program 1000
STATISTICS WHEN CLEANING THE SIMULATION:
    Removing packages...
        319 packages have been removed.
    Cleaning all stacks of packages...
        4 packages have been removed
    Cleaning shopping queue...
        22 robots have been removed
```