Hacettepe University
Department of Computer Engineering
BBM203 Programming Laboratory
Experiment 3

Subject : Car Production Line Simulation
Submission Date : 24.11.2016
Due Date : 11.12.2016
Programming Environment: ANSI C / MinGW / Windows
Advisors : Dr. Sevil ŞEN, Dr. Burcu CAN, Dr. Adnan ÖZSOY
Dr. Ahmet Selman BOZKIR

AIM

In this experiment you are expected to design and implement a basic console application that simulates a car production factory. The main goal of this assignment is to encourage students to employ linked list data structures by using dynamic memory allocation. Meanwhile, you will have a chance for reviewing and learning concepts such as pointers and queues.

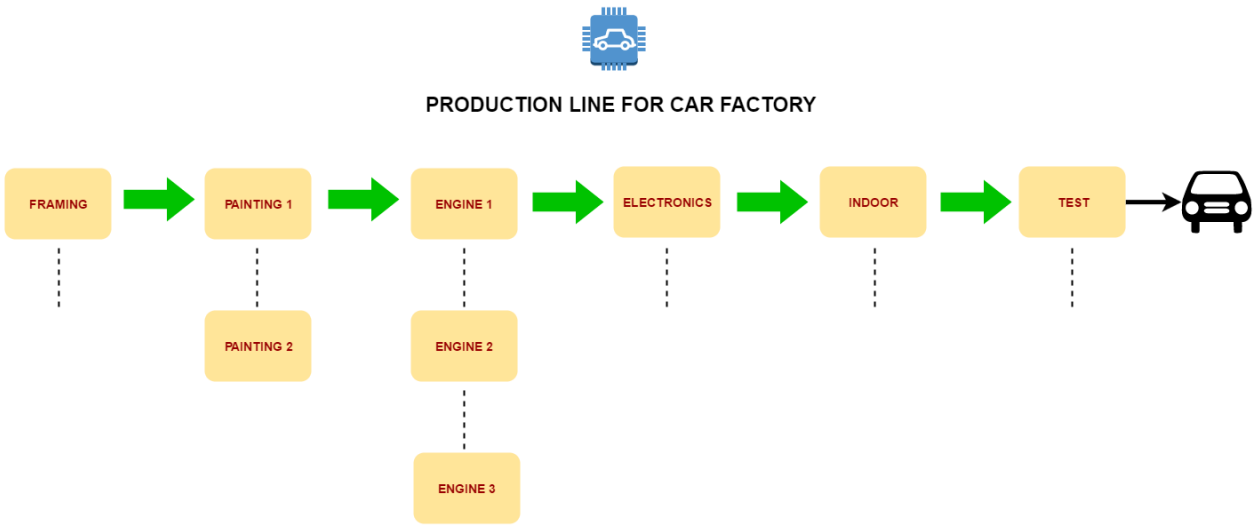
WARMUP

Production in factories is being handled by dividing whole process into different sub stages which are made by different departments. For instance, the process of car production involves several stages such as painting and engine assembly. On the other hand, for real world scenarios, these stages may be dependent or independent from each other. In other words, each stage may constitute a blocking factor to the next stage. In this regard, each stage and their sequential relations can be modelled by use of linked lists. In order to let you gain skills about how to create and manage linked lists, this homework's aim is to simulate a basic car production scheme.

PROCESS

As can be seen in figure below, in our hypothetical car factory (suppose that we are Volkswagen) you are supposed to produce arbitrary number of cars through arbitrary number of departments (each may have multiple sibling department with the same properties). All commands will be given via a text file (**text file name is variable** and will be supplied through command arguments) and output should be presented only on screen. Your application, meanwhile, must be implemented as a console application.

Actually, each car must visit different departments on the production line and needs a certain amount of time to be processed. Amount of required time varies in each department. For instance, framing may require 3T (*T here denotes a time span, i.e., minutes or hours*) or painting may need 4T. On the other hand, each car has a model name and a unique code (4 characters length). For example Golf G421 or Tiguan T632. According to start time, production process of each car begins. At each iteration (advancing to next T) each car is either processed (till to finish) or transferred to next department when its production stage at current department finishes. When a car is completely finished, it is removed from production line and placed at manufactured car list.



Note that, production start and transfer operations must be done by considering availability conditions of departments. Suppose that, we have two cars (their painting stage has been completed) and ready to transfer to engine department. However, if there exists only one available engine department you must put the first car to the available engine department and make the second one wait. Note that, you have to check the progress of departments in each iteration due to the different number of departments in each stage.

First, factory configuration will be supplied at input text file line by line in ordered format (details can be found at next section of this manual). Second, details of cars to be manufactured will be given in ascending start time ordered format. Third, report functions will be given as well as their parameters.

During your coding activities, you are **allowed** to use ANSI C libraries and syntax. All your codes must be collected in a single source file named "factory.c".

COMMANDS

Your application will involve 4 commands: **"AddDept", "Produce", "PrintFactory"** and **"Report".** Note that, **command names are case sensitive** and all commands will be error free in terms of syntax. Furthermore, delimiter character has been chosen as space character and all string values will be whitespace free (i.e., single word)

AddDept:

This command creates a department or an array of departments with supplied parameters. Command prototype is given below:

```
AddDept # of departments in that stage "Dept_Name" # of T
                      int                  string    int
```

Sample Command:

```
AddDept 2 Frame 3
Semantic:
Create two Frame departments that each can be completed in 3T
Sample Output:
Department "Frame" has been created
```

This sample command actually creates two departments with names “Frame1” and “Frame2” respectively. Moreover, “Frame2” will be the sibling of “Frame1”. Thus, an unsigned car will then be first tried to be assigned in Frame1. If this fails, Frame2 will be the next candidate.

Produce:
This command initiates the production of a car at specified T moment and outputs nothing. Command prototype is given below:

```
Produce T(Start Time) “CarModel” “Car Unique Code”
            int          string         string
```

Sample Command:
Produce 4 Golf G531
Semantic:
Start to manufacture a Golf with the code of G531 at 4T.
Sample Output:
N/A

The command of Produce actually adds a car to the list of cars that will be manufactured. In order to manufacture, your approach must check the list and transfer appropriate cars to the production line. You can use the unique code for different purposes.

PrintFactory:
This command takes no argument and prints the departments in a way that they have built. Simply, you are supposed to print the structure of your whole factory by placing each department type in a new line with indentation. Command prototype is given below:

Sample Command:
PrintFactory
Semantic:
Prints the hierarchical schema of the factory
Sample Output:

```
- Frame1 Frame2
            - Paint1 Paint2
                        - Engine1 Engine2 Engine3
                                                - Electronics1
                                                                - Indoor1 Indoor2 Indoor3
                                                                                        - Test1 Test2
```

Report:
In order to report actual status of a car or department at moment of T, “Report” command has been designed. Moreover, the “Report” command is being used to generate reports belonging to all cars and departments at a glance. As can be understood, it is a polymorphic command and supports 3 kinds of parameter types. It should be also noted that for any kind of reporting, you must print the command line before the execution of the reporting. You can see 4 different syntax examples below:

Sample Commands:
1) Report T Car “car unique code”
int string string
Semantic:
Print the status of the car with the id of G322 at the moment of T.
Sample Output:

```
Command: Car 9 G322
----------------------
|Report for Golf G322|
----------------------
Frame:2,  Paint:0,  Engine:0,  Electronics:0,  Indoor:0,  Test:0,  |  Start  Time:  7  |
Complete:12.5% | Not complete
```

In this particular example, at the 9th iteration, the car with id of "G322" has only been processed by department of "Frame". This is not surprising since it starts at 7th iteration and only 2T has been passed. Please note that the percentage of completeness has also been computed and printed along with Boolean result (i.e., complete or not complete)

2) Report T Cars
int string

Semantic:
Generate and print the status of all cars at the moment of T.
Sample Output:
Output will be in the same format shown in the previous version. However, you must print down all the cars instead of a single one.

3) Report T Departments
int string

Semantic:
Print the status of each department at the moment of T.
Sample Output:

```
Command: Departments 19
--------------------------------
|Report for Department "Test 1"|
--------------------------------
I am currently processing Jetta J041
--------------------------------
|Report for Department "Test 2"|
--------------------------------
Test 2 is now free.
----------------------------------
|Report for Department "Indoor 1"|
----------------------------------
Indoor 1 is now free.
Processed Cars
1. Golf G234
2. Jetta J041
----------------------------------
|Report for Department "Indoor 2"|
----------------------------------
I am currently processing Passat P862
Processed Cars
1. Touran T174
----------------------------------
|Report for Department "Indoor 3"|
----------------------------------
Indoor 3 is now free.
---------------------------------------
|Report for Department "Electronics 1"|
---------------------------------------
I am currently processing Golf G322
Processed Cars
1. Golf G234
2. Jetta J041
..
..
```

In this particular example, at the 19th iteration, a detailed report for all factory has been generated. Current status (i.e., whether it is free or processing) and list of the processed cars belonging to each department is being printed on screen.

REMARKS

- Transferring operation between departments must be designed by considering moving the car having the minimum start time. In other words, if there exist 3 cars to be moved to next department, your movement must carry the car having the least start time value first. Otherwise your algorithm will not work properly.
- Regardless of the command order, you can output the result of the "Report" commands by considering their T values in ascending order.
- Ensure that your source code is working prior to test stage. Crashing homeworks cannot exceed 30 points.

NOTES

- Input file will be in the same directory with your executable files. Therefore prepare your code to search the input file by its name (via argument) and process it.
- SAVE all your work until the experiment is graded.
- The assignment **must be original**, INDIVIDUAL work. Downloaded or modified source codes will be considered as cheating. Also students who share their work will be punished in the same way.
- You can ask your questions via course's piazza group. However, I suggest that you write your questions in English instead of Turkish.

SUBMISSONS

- The experiment code will be tested both MinGW and dev.cs.hacettepe.edu.tr. Please ensure that your code is well working and compliant with ANSI C standart.
- Your submission will be in the format below You must submit your code file in this format. For the process of obtaining the zip file, you can use Windows folder compression technique.

```
<BBM203_1617_3_StudentID>
        |-- source
                |-- factory.c
```

- Your code must be ready to be compiled by dev.cs.hacettepe.edu.tr GCC compiler and testing will be done by either dev or MinGW.
- Cheating is definitely prohibited. Students who are responsible for this kind of plagiarism will be punished by the rules of Hacettepe University. Therefore, if you want to cheat someone else code, think twice.
- You have to use "Online Experiment Submission System". **http://submit.cs.hacettepe.edu.tr** Other type of submissions especially by e-mail WILL NOT BE ACCEPTED.
- Submission deadline is 11.12.2016 23.59 pm. **No further extension will be given!!!**