Universidad de Alcalá

Laboratory Assignments

Data Structures

Spring 2022-2023

Terms and conditions:

Assignments should be done in groups of 2 people.

Assignments will consist in the creation of original C++ code, this means **written by the current students, in this course, and for this assignment**. Any copied code from any source could invalidate the whole effort of the students.

Students have to present in class every assignment to the teacher and the rest of the students and answer the teacher's questions about it on the corresponding lab session. You can find deadlines and days of presentation in the course presentation. These presentations should be supported with documents well suited for projector use.

Submissions that do not meet the deadlines and/or the requirements may not be reviewed. It is responsibility of the students to meet the requirements and it is not responsibility of the lecturer to warn students and go after them when such requirements are not met. Exceptions will not be considered (except for those included in the examination regulation-"*normativa de examenes*").

Students that do not pass the practical part of the course (i.e. the lab) on the January call may be required to complete a different assignment for the extraordinary call (also named Spring call or June call).

Important note: On these assignments only basic requirements are described. Students need to make additional assumptions and design decisions based on the given requirements. Students also need to decide on the input data and on the way in which that data is entered into the system. Sample data needs to be provided to facilitate testing. Students are free to implement the solution in the way they prefer. Any decision will be reasonable providing that (1) it is explained on the written document and commented on the source code, and (2) it does not contradict blatantly the requirements stated here.

Content to upload to BlackBoard Platform:

- Assignment shall be delivered as a GitHub repository shared with the teacher in the appropriate GitHub assignment.
- Assignment shall include the source files/projects as well as a document (.doc or PDF). The document must also include the IDs and names of all the members of the group.
- Assignment shall include the documentation used to support the explanations of the presentation day. PDF format is the only accepted format.

Documentation of every assignment

The documentation **must** contain at least the sections of the following index:

Index

1. Analysis
    1. Data flow from user or external files to system, internally in the system between every relevant component or storage, and from system to user or external files.
    2. ADT specifications:
        1. Explanation of the adaptations made to the standard specification of the ADT to fit requirements. If any ADT is used more than once, its description must be repeated for every case or stated to be the same than previous.
        2. Definition of operations of the ADT
            1. Name, arguments and return values

2. Design
    1. Diagram of the representation of the ADT in the memory of the computer (box diagrams) and explanation of every operation. Those explanations must be detailed only in case of operations that require it.
    2. Use case diagrams (UML notation)
    3. Class diagram
    4. Explanation of classes
        1. Explanation of significant methods. Including those created for operations and any other needed
    5. Explanation of the behavior of the program

3. Implementation
    1. Diagram with source files and their relations
    2. Explanation of every difficult section of the program

4. Review
    1. Running time of the solution (using big-Oh notation)
    2. Possible enhancements for the solution (in terms of efficiency and others)
    3. Reasoning on convenience or need of use of Dynamic DS in every case, comparing with Static DS characteristics.

5. References (Bibliography)

Common advice for all assignments

If you consider that any of these advice don't apply to your assignment, please answer lecturers to ensure it.

1. Students will provide the necessary files to test the programs if the requested program are able to manage them (more than one file and with different sizes). The number of elements must be enough to test all operations.

2. File management must not be part of the DS internal implementation.

3. The I/O operations, like printing and requesting data from user must not be part of any DS implementation.

4. The implementation of every assignment must not use high level libraries which implements DS and their operations.

5. The implementation of assignments must use dynamic memory storage (pointers) for dynamic DS implementation.

6. Creating a running program and a proper documentation, will result in a mark enough to pass the course. The better is the solution and the documentation, the higher will be the mark.

7. Failing to provide a working solution, an appropriate documentation or a clear presentation will result in low marks, not enough to pass the course.

8. Every program must produce enough output to show its internal behavior. Verbose programs are better, but the information must be organized to simplify the analysis, i.e. a list of 100 items, each one in a new line is worse than a list of items in 2-3 lines with some text explaining it.

Assignment: data structures and operations

Students are asked to implement an O.O. C++ program to process an image in the way explained in this proposal.

The program will execute three phases.

1. The first phase will be in charge of image loading, data extraction and data insertion in a data structure used to pass data the next phase.
2. The second phase will be in charge of data organization.
3. The second phase will be in charge of data exploitation, offering options to the user to manipulate data or to finish execution.

Image loading phase

The name of the image will be requested to the user. The program will only accept BMP images (https://es.wikipedia.org/wiki/Windows_bitmap).

For helping with image read and write a simple library is provided.

> The image read will be stored in a memory block without structure where you can access to any element using pointer arithmetic. The function used for reading will produce as output the width and height of the image. Every pixel will use 3 bytes to store the RGB value (Reg, Green and Blue).
>
> To access any pixel values, you will need to calculate the memory position using the following formulas (considering rows from 0 to width-1, and columns from 0 to height-1):
>
> - Red value: ImagePointer + (row * width) + column
> - Green value: ImagePointer + (row * width) + column +1
> - Blue value: ImagePointer + (row * width) + column +2

A queue storing all pixels (without filtering) will be created, named **QueA**.

Every pixel object will contain color and position information.

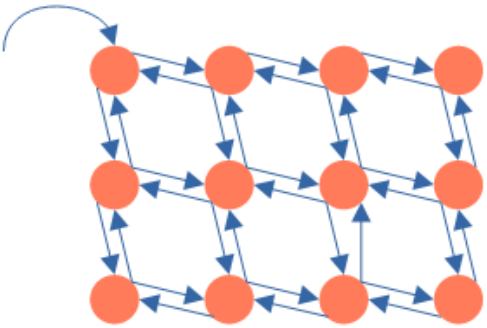
Data organization phase

In this phase, some versions of the storage data will be created, to test the running time of every solution.

Firstly, the program will extract every pixel object from the input queue (**QueA**) and generate:

1. A list will all elements, called **DaList** (not a data type, it is an instance of a list).

2. A mesh structure created to store the pixels. This structure will have a node for every pixel in the image and four links to adjacent pixels. The access pointer will be single or multiple as the students decide. This structure instance is called **DaPlace** (This name is for the explanation of actions, not a type or any internal name expected in the resulting program).



Using the data stored in both structures, some operations will be offered to the user in the next phase.

Data exploitation phase

In this phase, a menu is shown to allow the use of the data stored.

The options of the menu will be:

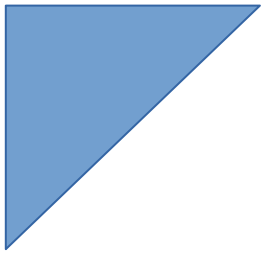
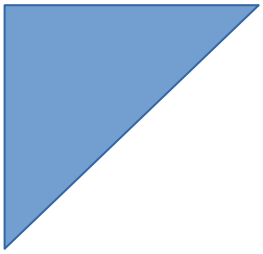
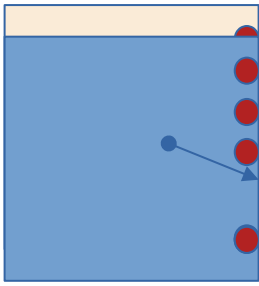
1. Search on the list.
2. Search on the mesh.
3. Mirroring using the mesh.
4. Mirror comparison using the mesh.
5. Exit

Explanation of the options of the menu:

1. **Search on the list.** This option will ask for the components of a pixel (R, G, B) and search all pixels with this components, creating a list with them, printing the time needed for the search, printing the list on the screen and then asking to the user for the new color for every one of this pixels that can be (255,0,0), (0,255,0) or (0,0,255). With the choose of the user, all found pixels will be changed and a new image will be saved with this changes. The original list must remain unchanged.

6. **Search on the mesh.** This option will ask for the components of a pixel (R, G, B) and search all pixels with this components, creating a list with them, printing the time needed for the search, printing the list on the screen and then asking to the user for the new color for every one of this pixels that can be (255,0,0), (0,255,0) or (0,0,255). With the choose of the user, all found pixels will be changed and a new image will be saved with this changes. The original mesh must remain unchanged.

7. **Mirroring using the mesh.** This one will allow to create a new image using all pixels on the left-upper triangle and duplicating them using the diagonal line as a mirror. A new image will be created on disk with the result and the original mesh must remain unchanged.

8. **Mirror comparison using the mesh.** In this option, the program will compare every pixel of the left-upper triangle with the counterpart in the right-lower triangle. The comparison of every pixel will get the absolute value of the differences of every component and get the average $(R_a-R_b+G_a-G_b+B_a-B_b)/3$. Finally, the average of the calculation of every pixel will be calculate and printed. The time for the calculations will be printed and the original mesh must remain unchanged.

9. **Exit**. Will ends the program.

The time of the search operations must be measured with different images (great differences in size) and appropriate images for testing will be provided, as well as resulting images after testing. A reasoning about the results must be presented.